Evaluación de modelos de aprendizaje profundo para la predicción de temperaturas máximas usando información espacio-temporal

Informe de Proyecto de Grado presentado por

Gonzalo Marco, María Eugenia Miranda

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Pablo Rodríguez Bocca

Montevideo, 29 de diciembre de 2023

Agradecimientos

Este proyecto marca el cierre de una etapa de nuestras vidas, ya que finalizamos la carrera de Ingeniería en Computación después de nueve años de esfuerzo. A lo largo de este tiempo hemos tenido la suerte de contar con el apoyo y la compañía de familiares y amigas/os. A ellos un profundo agradecimiento.

También queremos agradecer a Pablo Rodríguez-Bocca, nuestro tutor en este proyecto, por su colaboración, dedicación y orientación para que pudiera salir adelante.

Al equipo de *ClimateDL*, quienes nos han ayudado a darle un marco a este proyecto, a conseguir y mejorar los datos, y han colaborado en la escritura de nuestra primera publicación académica.

Finalmente, queremos expresar nuestro agradecimiento mutuo entre nosotros por el apoyo que nos hemos brindado todo este tiempo.

Quiero dedicarle este trabajo a mis abuelos, que
me enseñaron a no bajar los brazos.
Siempre los voy a tener presentes.
Gonza.

Resumen

En este proyecto evaluamos y comparamos dos modelos de aprendizaje profundo basados en paradigmas distintos, para descubrir su capacidad de inferir valores a base de la combinación de información espacio-temporal. Para ello, planteamos el objetivo de predecir temperaturas máximas a corto plazo en una amplia región del sur de América del Sur, usando información del pasado en conjunto con información geográfica. En la actualidad, para llevar adelante esta tarea se utilizan modelos físicos de predicción numérica, de alta complejidad, especificidad y costo en cómputo. Por otra parte, el área de aprendizaje automático ha logrado recientemente grandes avances en la predicción meteorológica.

Este proyecto se enmarca en el trabajo de un grupo interdisciplinario e internacional llamado *ClimateDL*, cuyo objetivo es evaluar distintos métodos para la predicción de temperaturas extremas a nivel estacional (mediano plazo con horizontes de varios meses) en el territorio sur de Sudamérica.

Para predecir temperaturas máximas en un horizonte de hasta 10 días, evaluamos dos modelos: *XGBoost* y *Graph WaveNet*. Como resultado de este trabajo, concluimos principalmente que *Graph WaveNet* resulta ser mejor que *XGBoost* para las predicciones a corto plazo. Y más importante aún, es que lo logra con mucho menos esfuerzo de acondicionamiento de datos. Por su parte, *XGBoost*, parece capturar mejor el comportamiento estacionario de la temperatura máxima, y, por lo tanto, su desempeño mejora comparativamente al aumentar el horizonte de observación. Además de los hallazgos mencionados, este proyecto ha contribuido al grupo *ClimateDL* mediante la presentación de resultados en una reunión presencial y la colaboración en la redacción de un artículo académico.

Palabras clave: Aprendizaje automático, series de tiempo, predicción de temperatura, información espacial, ventanas de tiempo, clima, meteorología, ClimateDL, XGBoost, Graph WaveNet

Índice general

Capítulo 1

Introducción

Existen muchos ejemplos de tareas de predicción basadas en series de tiempo, como predecir el precio de las acciones en los mercados financieros, pronosticar la demanda de productos, estimar el tráfico vehicular y prever la propagación de enfermedades. Sin embargo, esta tarea suele ser difícil de resolver (Chatfield, 2000). La predicción meteorológica es un ejemplo de estas tareas; y es de gran importancia en nuestra vida cotidiana, pero también en sectores productivos como la agricultura, el turismo, la energía, la anticipación de desastres naturales, entre otros.

El hecho de contar con pronósticos meteorológicos más precisos se ha convertido en una necesidad en un mundo donde el cambio climático afecta cada vez más y los eventos meteorológicos extremos son cada vez más frecuentes. Poder anticiparnos a diferentes escenarios o eventos nos permite prepararnos para ellos, salvando vidas, propiedades e incluso sociedades mediante la gestión de recursos. Un ejemplo de estos eventos es lo ocurrido en Uruguay durante la crisis hídrica en 2023[1]. Por lo tanto, trabajar en la obtención de técnicas aplicables a este desafiante problema climático es de gran valor para la sociedad. Es importante destacar que, aunque a veces usemos los conceptos de "meteorología" y "climatología" indistintamente, existen diferencias. La meteorología se enfoca en estudiar predicciones atmosféricas a corto plazo (días o semanas), mientras que la climatología busca patrones climáticos a largo plazo (años o décadas), buscando tendencias en el tiempo (Coleman y Law, 2015).

Ejemplos de pronóstico meteorológicos incluyen las temperaturas mínimas, medias y máximas en un lugar, día y altura específicos. Estos pronósticos tienen la particularidad de ofrecer predicciones a corto plazo, donde deseamos predecir datos en los próximos días. En la actualidad, estas predicciones se resuelven utilizando procesos físicos de granularidad fina, modelos con información de la atmósfera y los océanos (enormes sistemas de ecuaciones diferenciales, también conocidos como modelos dinámicos), ejecutados en potentes supercomputadoras. Estos modelos utilizan parámetros numéricos similares a los que se busca

[1] Informe BBC, https://www.bbc.com/mundo/articles/c4nvqjy9pywo, accedido el 02/09/2023.

predecir (se utiliza la temperatura del pasado para predecir la temperatura del futuro) y también parámetros relacionados con el contexto, como las precipitaciones y la humedad. Este proceso se denomina Predicción Numérica del Tiempo (NWP, por sus siglas en inglés, *Numerical Weather Prediction*) (Bauer, Thorpe, y Brunet, 2015). Ver (Granger y Joyeux, 1980) para profundizar en las técnicas estadísticas de pronóstico de series de tiempo basadas en NWP, especialmente para tareas de pronóstico a largo plazo.

En los últimos años, el Aprendizaje Automático ha experimentado un crecimiento significativo y ha realizado importantes contribuciones en diversas áreas, como la medicina, la industria automotriz, las finanzas, la publicidad, la traducción, la ciencia y la manufactura. Esto ha permitido avances en la investigación científica, en la optimización de la manufactura, y en la gestión de ciudades inteligentes, entre otros logros significativos. El área del pronóstico meteorológico no ha quedado al margen de esta revolución. Sin embargo, aún no se han obtenido resultados novedosos que superen a los modelos físicos existentes, ya sea para predicciones a corto, mediano o largo plazo. No obstante, existen modelos que se están acercando a los resultados de los modelos NWP, utilizando grandes recursos y conjuntos de datos (Lam y cols., 2022). Dentro del universo del Aprendizaje Automático, existen distintos paradigmas o arquitecturas, cada uno con ventajas y desventajas, y probablemente con una mayor aplicación en ciertas áreas específicas. Algunas de las arquitecturas tradicionales incluyen redes neuronales artificiales (RNN), máquinas de soporte vectorial (SVM) y árboles de decisión. Por otro lado, las arquitecturas basadas en grafos han ganado popularidad en los últimos años y se destacan por su capacidad para capturar relaciones complejas entre datos, lo que ha impulsado avances significativos en el campo del aprendizaje automático.

CLIMAT-AmSud, *STIC-AmSud* y *MATH-AmSud* son programas de colaboración científico-tecnológica entre los países Argentina, Bolivia, Brasil, Chile, Colombia, Ecuador, Paraguay, Perú, Uruguay, Venezuela y Francia. El objetivo es "*(...) promover y fortalecer la colaboración y la creación de redes de investigación-desarrollo en el ámbito de las ciencias y tecnologías de la información y comunicación (STIC), de las matemáticas (MATH) y del cambio climático y de la variabilidad climática (CLIMAT) a través de la realización de proyectos conjuntos*"[2]. Uno de los proyectos financiados por *CLIMAT-AmSud* es *ClimateDL*[3], iniciado en 2021, es un trabajo colaborativo entre investigadores de diversas áreas (climatología, ciencia de datos y matemáticas) de distintos países sobre la predicción de temperaturas extremas a escala estacional (horizonte de varios meses) en el sur de América del Sur mediante métodos de aprendizaje automático, aprendizaje profundo y métodos estadísticos, utilizando en lo posible, técnicas de análisis de redes. Nuestro trabajo está enmarcado en dicho proyecto *ClimateDL*.

En nuestro trabajo tenemos como objetivo principal comparar modelos de aprendizaje profundo para pronosticar las temperaturas máximas diarias en un

[2] https://www.sticmathamsud.org/sitio/, accedido el 02/08/2023

[3] (22-CLIMAT-02) https://climatedl.pages.fing.edu.uy/website/about.html, accedido el 02/08/2023

horizonte de hasta 10 días, en la región sudamericana, que incluye Argentina, Brasil, Chile, Paraguay y Uruguay. Estos pronósticos se realizarán en un amplio conjunto predeterminado de puntos geográficos, donde se encuentran estaciones de medidas meteorológicas, de donde se obtienen los datos. Compararemos dos clases de modelos de aprendizaje, los métodos basados en *gradient boosting trees* y los basados en *graph neural networks*, eligiendo un representante de cada clase. Para llevar a cabo esta tarea necesitamos plantearnos el objetivo específico de generar un conjunto de datos que concentre la información meteorológica de la región. La centralización de esta información es un resultado valioso, ya que hoy en día no es posible obtenerla de forma directa. Luego, implementaremos un modelo moderno de la clase *gradient boosting trees*, llamado *XGBoost* (Chen y Guestrin, 2016), que requiere cierto preprocesamiento de datos y ha demostrado buenos resultados en diversas áreas, incluyendo problemas de series temporales. Por otro lado, en el ámbito de los modelos de aprendizaje automático basados en grafos, utilizaremos la arquitectura conocida como *Graph WaveNet* (Wu, Pan, Long, Jiang, y Zhang, 2019), que se ha destacado por el uso de una matriz de adyacencia para modelar las relaciones espaciales entre las ubicaciones geográficas del conjunto de datos. Finalmente, compararemos estos modelos para evaluar su rendimiento en términos de precisión en las predicciones, complejidad de gestión de datos y tiempos de ejecución. Buscamos determinar si *Graph WaveNet* logra obtener mejores resultados que *XGBoost* en la predicción meteorológica, en particular evaluaremos si logra capturar automáticamente la interrelación espacial existente en los datos entre diferentes ubicaciones geográficas, algo que *XGBoost* requiere que sea realizado de forma manual y habitualmente insume mucho tiempo y trabajo por parte de expertos en el problema predictivo.

En el informe, podrán observar que hemos aplicado ambas técnicas, logrando que los modelos obtengan resultados aceptables y verificando nuestra hipótesis inicial con respecto a las ventajas de *performance* de los modelos basados en grafos. Además, como resultado adicional, hemos contribuido al proyecto *ClimateDL*, presentando nuestros hallazgos en un encuentro realizado por los integrantes del proyecto, que tuvo lugar en Montevideo en noviembre de 2022[4]; y también hemos contribuido en la redacción de una publicación académica en el *2nd International Workshop on Machine Learning for Irregular Time Series (ML4ITS2023): Advances in Generative Models, Global Models and Self-Supervised Learning*[5] parte de la prestigiosa conferencia *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, nuestro trabajo titulado "Maximal Temperature forecasting under spatio-temporal interrelations using Machine Learning" resume en gran medida este documento y nuestro aporte al proyecto (Marco, Miranda, Rodríguez-Bocca, y Rubino, 2023).

Como podemos apreciar, este trabajo tiene un enfoque de investigación, por lo que no buscamos obtener un producto final. Este documento describirá el proceso desarrollado para alcanzar el objetivo establecido.

[4] https://climatedl.pages.fing.edu.uy/website/workshop.html, accedido 2023-10-08
[5] https://ml4its.github.io/ml4its2023, accedido en 2023-10-08.

En el Capítulo 2 resumimos el marco teórico relacionado con la temática. Además, proporcionamos un resumen del estado del arte en este campo y analizamos los modelos propuestos, destacando sus características, funcionamiento, ventajas y desventajas. En el Capítulo 3, ofrecemos una descripción detallada del trabajo relacionado con la obtención, análisis y gestión de los datos. Luego, explicamos la metodología del trabajo de manera general, junto con comentarios específicos para cada modelo abordado. El Capítulo 4 resume las experimentaciones llevadas a cabo y presenta los resultados obtenidos. Finalmente, en el quinto y último capítulo, se presentan las conclusiones del proyecto y se discute sobre el posible trabajo futuro relacionado con el proyecto.

Capítulo 2

Revisión de antecedentes

En este capítulo formalizamos el problema a resolver (Secciones 2.1 y 2.2) planteando la función objetivo a minimizar bajo el enfoque de series de tiempo y espacio geográfico. En la Sección 2.3 presentamos los trabajos recientes en esta área que han logrado buenos resultados. Adicionalmente, en las Secciones 2.4 y 2.5 presentamos los modelos de aprendizaje profundo utilizados en este trabajo.

2.1. Definición del problema

Buscamos predecir la temperatura máxima de los siguientes días en un conjunto de puntos geográficos, basándonos en datos históricos que incluyen características meteorológicas como temperaturas y precipitaciones, junto con información geográfica relevante. Para lograr esto, es esencial comprender cómo manejamos la información de entrada, que consiste en mediciones similares pero tomadas en el pasado en el mismo lugar.

Formalmente, podemos formular el problema de pronóstico de la siguiente manera. Tenemos a $\mathcal{P}$, el conjunto de N puntos geográficos de interés de los cuales queremos predecir sus temperaturas máximas en el futuro próximo. Conocemos los datos históricos de las temperaturas máximas en esos puntos para distintos pasos (discretos) de tiempo $t \in \mathbb{N}$, simbolizado con $\mathbf{y}_t \in \mathbb{R}^N$, y otros posibles atributos predictores (*features*), simbolizados con $\mathbf{X}_t \in \mathbb{R}^{N \times D}$ (N puntos con D atributos predictores). Por lo tanto, podemos definir nuestro problema como la búsqueda de una función modelo $f()$, capaz de predecir la temperatura máxima de los próximos S pasos, en función de los valores históricos de temperaturas y atributos de los T pasos anteriores:

$$\mathbf{y}_{(t-(T-1):t}, \mathbf{X}_{(t-(T-1)):t} \xrightarrow{f} \mathbf{y}_{(t+1):(t+S)}, \tag{2.1}$$

donde $\mathbf{y}_{(t-(T-1):t} \in \mathbb{R}^{N \times T}$, $\mathbf{X}_{(t-(T-1)):t} \in \mathbb{R}^{N \times D \times T}$ y $\mathbf{y}_{(t+1):(t+S)} \in \mathbb{R}^{N \times S}$.

En nuestro caso, los puntos geográficos corresponden a estaciones de medición meteorológica, el paso de tiempo es un día, y el problema más sencillo que podemos atacar corresponde a predecir la temperatura máxima del siguiente

día ($S = 1$), a partir de exclusivamente las temperaturas máximas del pasado reciente (por ejemplo los 12 días anteriores, $T = 12$), es decir, sin contar con atributos predictores extras ($D = 0$), el problema se simplifica en:

$$\mathbf{y}_{(t-12):t} \xrightarrow{f} \mathbf{y}_{t+1}, \tag{2.2}$$

donde $\mathbf{y}_{(t-12):t} \in \mathbb{R}^{N \times 12}$ y $\mathbf{y}_{t+1} \in \mathbb{R}^{N}$.

La Figura 2.1 representa lo explicado antes, con un conjunto de entrada de 12 días para predecir el día siguiente (T=12, S=1).

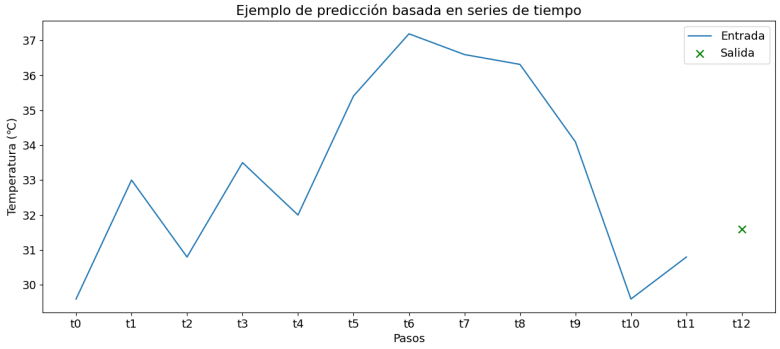

Figura 2.1: Ejemplo de predicción temporal del día siguiente.

Como medida para evaluar qué tan buenas son las predicciones en una fecha específica, usamos el error cuadrático medio ($RMSE$):

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{2.3}$$

donde, en nuestro caso, N es el número de estaciones meteorológicas, $\mathbf{y} \in \mathbb{R}^N$ es el vector objetivo de temperaturas y $\hat{\mathbf{y}} \in \mathbb{R}^N$ es el vector de predicción resultante del modelo $f()$. Esta fórmula se aplica para la medición de una fecha (t) en particular.

2.2. Modelos para pronósticos espacio-temporales

2.2.1. Predicción temporal con series de tiempo

En general, una predicción es el resultado de un pronóstico a base de un conjunto de datos dado. Se suelen usar modelos de inteligencia artificial entrenados con estos conjuntos de datos para obtener el pronóstico. Las predicciones pueden ser de eventos a suceder en el futuro (valor de la moneda, registros climáticos), o también respecto a hechos ya sucedidos (detección de fraudes, interés de compra, cantidad de lluvia en puntos no observados).

El conjunto de datos usado como entrada para realizar la predicción es información relacionada con el resultado objetivo. Cuando se busca realizar una predicción en el futuro, este conjunto de datos puede ser una serie de tiempo. Una serie de tiempo es una secuencia de registros (datos observados) sobre el mismo objetivo que es registrado con el paso del tiempo.

Un concepto clave en lo que respecta a la predicción basada en series de tiempo es el de la ventana de tiempo. El mismo refiere al número de pasos de tiempo anteriores que se usan como entrada al modelo para hacer la predicción deseada. Usualmente, la información en la ventana es del mismo tipo que la que se quiere predecir (se maneja temperaturas del pasado para predecir temperaturas del futuro), pero también puede llegar a contener otros datos relacionados. Si bien este concepto se usa principalmente para la información usada como entrada, también se puede referir a la ventana de tiempo de salida: en casos donde el problema busca predecir más de un paso en el futuro. Existen dos posibles estrategias al momento de procesar los datos para luego poder entrenar y generar predicciones: 'ventana deslizante' y 'ventana expandible' (Markudova y cols., 2019). La primera maneja un tamaño fijo de los datos de entrada y 'se desliza' a medida se avanza en el tiempo, mientras que la segunda maneja un tamaño variable de la ventana de entrada donde en general suele acumular datos a medida se avanza. La Figura 2.2 muestra un esquema de cada una de ellas. El uso de una técnica u otra dependerá de las características del proyecto y del poder de cómputo disponible.

La elección del tamaño de la ventana de tiempo influye en la precisión de los resultados del modelo: Una ventana pequeña puede llegar a no capturar toda la información necesaria y una muy grande puede sobre ajustarse, aunque también suelen capturar patrones más complejos.

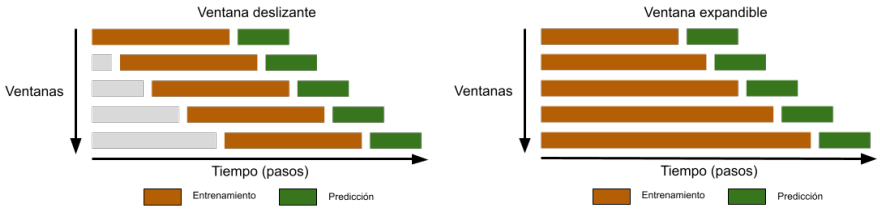

Figura 2.2: Ejemplos de las estrategias 'ventana deslizante' y 'ventana expandible' para el manejo de ventanas de tiempo en las predicciones basadas en series temporales.

Según (Lim y Zohren, 2021), las predicciones basadas en series de tiempo son modelos que predicen un registro $y_{i,t}$ a ocurrir en el futuro, para una entidad i en un momento t del tiempo. Cada entidad representa un grupo lógico de información temporal (por ejemplo, la medición histórica de temperaturas mínimas, temperaturas máximas y de precipitaciones) la cual puede ser manejada en conjunto.

La implementación de modelos para problemas de series de tiempo es amplia

y variada, donde se incluyen las técnicas predictivas tradicionales[1]. Ejemplos típicos son el promedio de los registros de entrada, interpolación exponencial y ARIMA, u otros modelos como Redes Neuronales, Transformers y *XGBoost*. La calidad de estos modelos es evaluada habitualmente usando medidas como Mean Square Error (*MSE*) y Root Mean Square Error (*RMSE*).

Se pueden encontrar varios problemas de predicciones temporales de series de tiempo multivariables (tráfico, clima, cotizaciones). En (Onur Bilgin and Pawel Maka and Thomas Vergutz and Siamak Mehrkanoon, 2021) se indica que muchos estudios se encuentran investigando para encontrar nuevas tendencias y patrones a base de series de tiempo basado en el historial registrado en el área. En particular, en los últimos años se ha registrado un avance en el desarrollo de técnicas de aprendizaje profundo en lo que respecta a las predicciones basadas en series de tiempo.

2.2.2. Predicción espacio-temporal basada en grafos

En la última década, se ha consolidado el uso del aprendizaje profundo (*deep learning - DL*), basado en redes neuronales, para distintos problemas predictivos. En especial, por lo general, el aprendizaje profundo es la solución más competitiva para problemas donde las muestras de datos son (suficientemente) independientes. En los últimos años, estas técnicas se han comenzado a extender a problemas con datos altamente interrelacionados (como ocurre al combinar el espacio y el tiempo). En este contexto, surgen los métodos de aprendizaje profundo basados en grafos, en donde se crean estructuras de redes neuronales orientadas a grafos. Según (Wu y cols., 2019), "Una suposición básica detrás del modelado espacio-temporal basado en grafos es que la información futura de un nodo del grafo está condicionada por su información histórica y por la información histórica de sus vecinos. Por lo tanto, capturar las dependencias espaciales y temporales se convierte en un desafío primordial."

La predicción espacio-temporal basada en grafos es un método que se basa en la teoría de grafos para representar, analizar y detectar relaciones entre variables temporales y espaciales con las cuales puede producir predicciones relacionadas con la información histórica brindada. En general, los nodos del grafo representan puntos espaciales, y las aristas la información que las relaciona. Mientras que los atributos de los nodos son dinámicos, es decir, varían con el tiempo. El uso de estas técnicas de predicción ha aumentado últimamente y se pueden ver aplicaciones en análisis de tráfico (Jiang y Luo, 2022; Wu y cols., 2019), de clima (Keisler, 2022; M. Ma y cols., 2023; Lam y cols., 2022), o de otras características viales de una ciudad (J. Ma, Chan, Rajasegarar, y Leckie, 2022) (Li, Zhou, y Pan, 2022).

[1] https://paperswithcode.com/task/time-series-forecasting, accedido el 20-03-2023

2.3. Estado del arte

En esta Sección procedemos a exponer el análisis de trabajos actuales con similitudes al propuesto. Para la selección de los mismos buscamos que sean recientes, con un enfoque de predicción meteorológica basada en series de tiempo y que usen arquitecturas de aprendizaje profundo.

Estos informes han sido de gran valor, ya que fueron una referencia durante nuestro proceso de investigación y experimentación.

2.3.1. Forecasting The Air Temperature at a Weather Station Using Deep Neural Networks

El artículo (Roy, 2020) resume la evaluación de tres arquitecturas tradicionales de redes neuronales, entrenadas para la predicción de la temperatura promedio de una sola estación (punto geográfico). Las tres arquitecturas reciben información meteorológica de una serie de tiempo y buscan la predicción del día siguiente y además de varios días en conjunto.

El conjunto de datos usado consta de diez años (del 1/1/2009 al 1/1/2019) de mediciones diarias en el aeropuerto internacional John F. Kennedy. Cada registro contiene información sobre: velocidad promedio del viento, precipitaciones, nevadas, profundidad de la nevada, temperatura promedio, temperatura máxima y temperatura mínima. El mismo fue dividido en subconjuntos para realizar distintas tareas. Los primeros 8 años (80 % del total) fue reservado para entrenamiento, mientras que los 2 años finales (20 % del total) para testeo.

Todos los modelos fueron definidos para recibir como entrada una serie de tiempo de 7 días consecutivos, y devolver, por un lado, la predicción de la temperatura promedio del día siguiente, y además la predicción en conjunto de los siguientes 10 días.

Los modelos tradicionales implementados para la experimentación fueron los siguientes:

- **MLP (Multi-Layer Perceptron)**(Ramchoun, Amine, Idrissi, Ghanou, y Ettaouil, 2016): es una red *feed-forward* (Svozil, Kvasnicka, y Pospichal, 1997; Cruz y cols., 2007) (redes sin ciclos, las capas de un nivel solo reciben información del anterior) que logra identificar relaciones complejas en los datos. Contiene una capa de entrada, capas ocultas, y una capa de salida. Este modelo se puede aplicar para clasificación y regresión. El texto del trabajo indica que esta arquitectura puede sufrir falta o sobre entrenamiento si la cantidad de neuronas no se configura correctamente al problema y datos brindado.

- **LSTM (Long Short Term Memory Network)**(Hochreiter y Schmidhuber, 1997): es una red neuronal recurrente (conjunto de sistemas dinámicos que es capaz de guardar información de pasos anteriores), por lo cual es capaz de mantener memoria de la información gracias a su arquitectura cíclica en las capas ocultas. Esto es de utilidad para predicciones

de series de tiempo, pero es sabido que no logra desempeñarse muy bien para largos períodos. Requiere más memoria que el modelo MLP.

- **CNN + LSTM (Convolutional Neural Network + LSTM)**: la arquitectura CNN (Gu y cols., 2018) (redes convolucionales) logra extraer las características, patrones en cortos períodos y relaciones entre las mismas. La salida del CNN es la entrada del LSTM. Esta concatenación resultante muestra buenos resultados en distintas áreas como reconocimiento de voz y predicción en series de tiempo.

Para medir y comparar el rendimiento de los distintos modelos se usaron las medidas *RMSE* y *MAPE*. La medida *Mean Absolute Percentage Error* (*MAPE*), indicada en la Ecuación (2.4), refleja en forma porcentual el error entre los valores predichos (y_i) y los esperados ($\hat{y}_i$). Si el porcentaje se encuentra próximo al 100 % implica que los valores predichos se encuentran más próximos al valor medido.

$$\text{MAPE} = \frac{1}{N} \sum_{i=1}^{N_{\text{pred}}} \left| \frac{y_i - \hat{y}_i}{\hat{y}_i} \right|. \tag{2.4}$$

En el Cuadro 2.1 se muestran los resultados de evaluar cada uno de los modelos entrenados sobre el conjunto de datos de testeo. Allí se puede observar que el modelo con la arquitectura *CNN + LSTM* fue el que se desempeñó mejor en la predicción del día siguiente y del conjunto de 10 días. Luego el modelo *LSTM* resultó superior al *MLP*.

	1 día		**10 días**	
	RMSE	MAPE	RMSE	MAPE
MLP	0.2	89.15 %	0.45	63.23 %
LSTM	0.1	95.03 %	0.4	70.32 %
CNN+LSTM	0.075	97.42 %	0.38	71.58 %

Cuadro 2.1: Comparación de métricas obtenidas en los distintos modelos al evaluar solo el siguiente día y el conjunto de los siguientes 10.

Más allá de que algún modelo tuvo mejor rendimiento que los otros, es importante notar que todos los planteados logran resolver el problema de buena forma, con una medida de error en general de menos que un grado; y además se puede notar que la tarea de predecir solo el día siguiente es un problema más sencillo de resolver que intentar predecir más de un día a la vez. Los autores del artículo indican además que la performance de los modelos se puede mejorar si se construyen redes neuronales más complejas (por ejemplo, aumentando la cantidad de capas o de nodos).

Otro detalle a tener en cuenta respecto a los registros de las métricas es el tamaño del conjunto de datos. Se manejaron en total diez años de mediciones, y solo los dos últimos para testeo, lo que implica poca variabilidad de los registros, además de un muestreo pequeño.

Adicionalmente, es importante destacar que los investigadores se centraron en el problema de predicción en un único punto geográfico, a diferencia de nuestro problema con varias locaciones.

2.3.2. Purely data-driven medium-range weather forecasting achieves comparable skill to physical models at similar resolution

Este trabajo (Rasp y Thuerey, 2021) busca la predicción de distintos datos meteorológicos como temperaturas y precipitaciones hasta cinco días en el futuro. Para ello se manejó un conjunto de registros históricos con información meteorológica de distintas estaciones de medición, y modelos basados en arquitecturas tradicionales.

El conjunto de datos es WeatherBench[2] (Rasp y cols., 2020), una fuente de datos conocida por su tamaño y usada en este tipo de tareas, ya que es el resultado de un procesamiento del conjunto de datos *ERA5* (Hersbach y cols., 2020) (un conocido conjunto de datos con datos meteorológicos a nivel mundial). El conjunto de datos usado contiene registros de 40 años en 2048 estaciones, y en cada uno de estos puntos de medición se tienen registros por hora con información variada del siguiente tipo: temperaturas (con mediciones en distintas alturas), viento, humedad (con mediciones en distintas alturas) y precipitaciones. Del conjunto total de datos, solo los últimos dos años fueron reservados para tareas de evaluación y testeo (es decir, el 5 % del total).

Para la experimentación, se planteó utilizar la información meteorológica (mencionada anteriormente) de la hora actual, 6 y 12 horas anteriores. Puntualmente, la información meteorológica utilizada fue:

- *500 hPa geopotential* (*Z500*, información atmosférica medida en $m^2 s^2$);
- *850 hPa temperature* (*T850*, información atmosférica medida en grados Kelvin); y
- *2-meter temperature* (*T2M*, temperatura medida en grados Kelvin),
- la acumulación de precipitaciones de 6 horas (*PR*, medida en mm).

En este caso no se buscó la predicción de un día en particular, sino que de distintos conjuntos. En particular, predecir el conjunto de los siguientes tres días y el conjunto de los siguientes cinco.

Para esta investigación, se desarrolló una variante de modelos convolucionales llamada *Resnet* (He, Zhang, Ren, y Sun, 2016), en donde se crea un modelo para cada parámetro que se quiere predecir. A su vez, se aplicaron dos técnicas para entrenar cada uno de estos. Por un lado, la llamada "directa"donde se entrena un modelo para cada fecha a predecir (es decir, que se crean 3 modelos para predecir 3 fechas), y luego una técnica "continua", la cual se entrena un

[2]WeatherBench: https://github.com/pangeo-data/WeatherBench, accedido el 01-12-2022

único modelo y se agrega como entrada la fecha que se quiere predecir. Adicionalmente, se menciona que existe una tercer técnica llamada "iterativa", la cual genera la predicción para la menor unidad de tiempo posible y dicho resultado es usado como entrada para predecir el siguiente paso. Este proceso se repite hasta alcanzar el o los registros temporales buscados.

A su vez se realizaron las mismas predicciones del conjunto de datos con métodos físicos (técnicas usadas hoy en día, que usan una aproximación a base de fórmulas físico-matemáticas) de utilidad para comparar los rendimientos con una metodología usada hoy en día, en particular se hace hincapié en la técnica llamada *Integrated Forecasting System* (IFS) [3].

Para medir y comparar el rendimiento de los distintos modelos se usó la medida *RMSE*, con cierta diferencia en la fórmula usual, ya que aplica una ponderación sobre la ubicación geográfica de cada estación de medición. Las Ecuaciones 2.5 y 2.6 reflejan cómo se toma en cuenta la latitud y longitud para comparar el registro predicho frente al esperado. A su vez, la fórmula toma en cuenta las múltiples fechas a predecir para devolver una única medida.

$$\text{RMSE} = \frac{1}{N_{\text{pred}}} \sum_{i=1}^{N_{\text{pred}}} \sqrt{\frac{1}{N_{\text{lat}} N_{\text{lon}}} \sum_{j}^{N_{\text{lat}}} \sum_{k}^{N_{\text{lon}}} L(j) \frac{y_{i,j,k} - \hat{y}_{i,j,k}}{\hat{y}_i}}, \tag{2.5}$$

$$\text{L(j)} = \frac{cos(\text{lat}_j)}{\frac{1}{N_{\text{lat}}} \sum_{j}^{N_{\text{lat}}} cos(\text{lat}_j)}. \tag{2.6}$$

Como resultados de la experimentación en el Cuadro 2.2 se observa que los modelos creados tienen un error medio bastante cercano al que maneja el modelo físico. Sin embargo, no lograron desempeñarse mejor que este, a excepción del caso de las precipitaciones, en donde estos lo superan levemente. A su vez, se puede observar que los modelos directos se desempeñaron mejor que los continuos para la predicción de tres días, y sucedió lo opuesto para las predicciones de cinco días.

	Z500		**T850**		**T2M**		**PR**	
	3 días	**5 días**	**3 días**	**5 días**	**3 días**	**5 días**	**3 días**	**5 días**
IFS	**154**	**334**	**1.36**	**2.03**	**1.35**	**1.77**	2.36	2.59
Directo	268	523	1.65	2.52	1.42	2.03	**2.16**	**2.30**
Continuo	284	499	1.72	2.41	1.48	1.92	2.23	2.33

Cuadro 2.2: Comparación de métricas RMSE obtenidas al evaluar modelos directos, continuos e IFS.

No se encontraron registros que indiquen de forma explícita los resultados de las predicciones para el día siguiente, como se tiene para 3 y 5 días. Sin embargo, en la Figura 2.3 (obtenida del artículo original) puede apreciarse el

[3] https://www.ecmwf.int/en/research/modelling-and-prediction, accedido el 01-12-2022

comportamiento de los modelos en la predicción de cada paso de la ventana de tiempo. Particularmente se puede observar en la Figura (c) que el modelo directo obtiene mejores resultados que el físico para el siguiente día, ambos levemente por debajo de 1,0 °K.

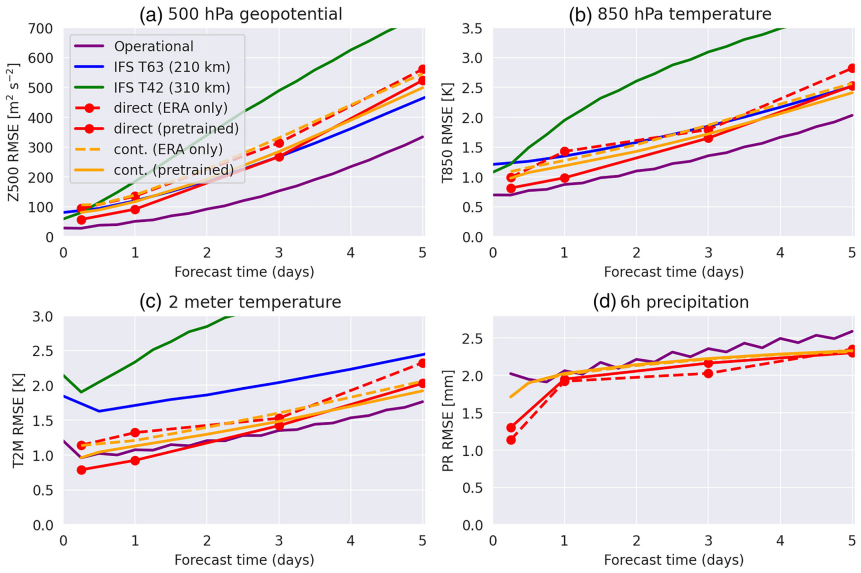

Figura 2.3: RMSE de los modelos entrenados y el modelo físico IFS para las distintas predicciones. Ilustración obtenida del artículo original (Rasp y Thuerey, 2021).

Este trabajo agrega la dimensión del espacio al problema de predicción meteorológica y presenta modelos que logran resolver el mismo de forma satisfactoria, con un margen de error similar a uno de los modelos usados en la actualidad. El origen de los datos es de una fuente confiable y de gran tamaño, se maneja una cantidad considerable de parámetros registrados por hora. Todos estos factores resultan de gran valor al momento de buscar resolver el problema. Como se verá más adelante, en nuestro caso no se logra esta granularidad en los datos, lo que puede afectar los resultados finales. Adicionalmente, aunque nuestro trabajo hará mención de la predicción de un conjunto de días en el futuro, como se hace en este, se enfocará principalmente en la predicción de uno solo.

2.3.3. TENT: Tensorized Encoder Transformer for temperature forecasting

TENT (Onur Bilgin and Pawel Maka and Thomas Vergutz and Siamak Mehrkanoon, 2021) es otro ejemplo de un modelo capaz de realizar predicciones a base de información espacio-temporal. Se hacen predicciones de temperaturas hacia el futuro, teniendo información del histórico de las mismas en distintos

puntos geográficos. Para ello se representa toda la información con tensores: estructuras de datos de varias dimensiones, donde cada una de estas refiere a un tipo de parámetro. En este caso se manejan tensores de 3 dimensiones (fecha, ubicación, parámetro de medición).

Los datos usados provienen de dos conjuntos distintos y se manejan en experimentos separados. Por un lado, se tiene 5 años de registros en 30 ubicaciones en Estados Unidos y Canadá, con mediciones por hora y 11 parámetros meteorológicos. Por otra parte, se tienen registros de Europa con mediciones diarias por un período de 15 años y 19 ubicaciones, donde se manejan 19 parámetros para cada registro. Ambos conjuntos de datos fueron divididos de igual forma: el primer 80 % del período fue usado para entrenamiento, luego un 2 % para validación y el restante 18 % para testeo.

Con los registros meteorológicos y la ubicación geográfica de las estaciones se busca predecir temperaturas promedio para distintos registros en el futuro. Debido a que los conjuntos manejan distintas unidades de medida del tiempo (uno en horas y el otro en días) y poseen datos de distintos espacios geográficos, se realiza un experimento particular para cada una de las fuentes de datos. En el caso de Estados Unidos y Canadá, se usa una ventana de tiempo de 16 horas para predecir las siguientes 4, 8, 12 y 16 horas; cada una de las predicciones es realizada de forma independiente de las otras. Para el caso de los registros de Europa se ingresan los últimos 8 días de información para predecir la temperatura media dentro de 2, 4 y 6 días; y al igual que en el primer conjunto, estas predicciones se entrenaron y realizaron de forma independiente.

El modelo construido se basa en la arquitectura convencional de un *Transformer* (Vaswani y cols., 2017) (arquitectura con capas de activación y *feed forward*), a la que se le agrega a su principio una capa llamada *encoding* (alteración de los valores, rangos y dimensiones del conjunto para adaptarlo a otro formato, entendible por la arquitectura), que se enfoca en las variables meteorológicas (relacionando y marcando dependencias) y persiste la información de tiempo y ubicación.

Se aplicaron otros modelos de aprendizaje profundo a los mismos problemas objetivo, de modo de tener comparaciones con arquitecturas que ya se conoce su funcionamiento: *Transformer* clásico, *3D CNN* (Mehrkanoon, 2019), *LSTM* (Hochreiter y Schmidhuber, 1997) y *ConvLSTM* (SHI y cols., 2015). Todos estos modelos y el modelo planteado *TENT* fueron entrenados con los dos conjuntos de datos y se registraron los resultados.

Para medir y comparar el rendimiento de los distintos modelos se usaron las métricas *MAE* y *MSE*. La métrica *Mean Absolute Error* (*MAE*, por sus siglas en inglés) refleja el promedio de la diferencia entre los valores medidos y los esperados, que en este caso miden la diferencia en forma absoluta registrada entre las predicciones y las mediciones reales (ver Ecuación 2.7). Por otra parte, el error cuadrático medio (*MSE* o *Mean Squared Error*) mide el promedio de los errores al cuadrado entre los valores predichos y los esperados. Su fórmula, reflejada en la Ecuación (2.8), nota una gran similitud con la métrica *RMSE* comentada previamente. En ambas fórmulas y_i refiere a un valor esperado, $\hat{y}_i$ al estimado en un punto geográfico, y N es la cantidad de estaciones de medición.

$$\text{MAE} = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N}. \tag{2.7}$$

$$\text{MSE} = \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}. \tag{2.8}$$

Las Figuras 2.4 y 2.5 reflejan las métricas *MAE* para el conjunto de datos de Estados Unidos y Canadá, y el conjunto de datos europeo, respectivamente. Los resultados reflejan que el modelo *TENT* entrenado para el primer conjunto se desempeñó mejor que los demás al usar los datos de evaluación, mientras que el correspondiente al conjunto de datos de Europa superó a todos los modelos, excepto el que es basado en una *LSTM*.

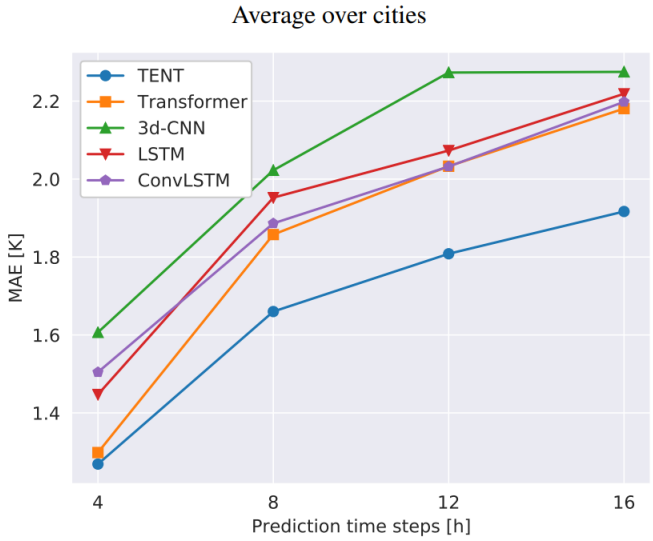

Figura 2.4: Promedio de métricas *MAE* para los modelos al evaluar sobre los datos de Estados Unidos y Canadá. Ilustración obtenida del artículo original (Onur Bilgin and Pawel Maka and Thomas Vergutz and Siamak Mehrkanoon, 2021).

El trabajo plantea un problema objetivo bastante similar al propuesto por nosotros y además maneja parámetros similares. Sin embargo, nuestro caso puede presentar mayor variabilidad de los registros debido a la diversidad geográfica que se nos plantea. Adicionalmente, si bien la estructura de datos manejada como tensores es diferente, el planteamiento del modelo y como interpreta la información en conjunto presenta similitudes a como lo hace *Graph WaveNet* (arquitectura utilizada en nuestro proyecto).

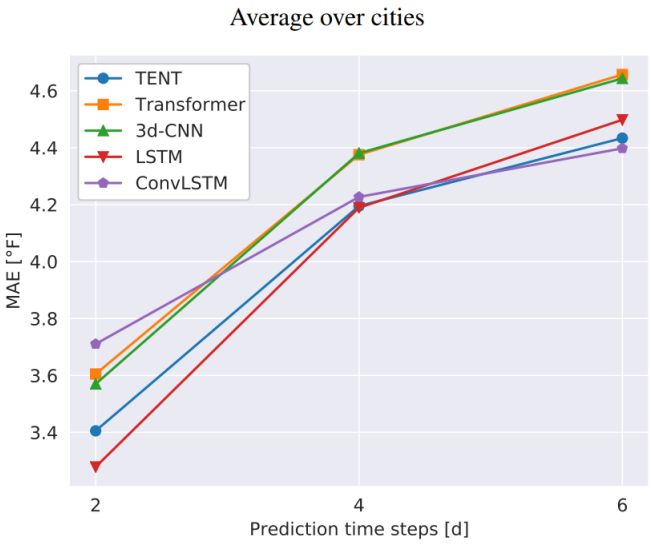

Figura 2.5: Promedio de métricas *MAE* para los modelos al evaluar sobre los datos de Europa. Ilustración obtenida del artículo original (Onur Bilgin and Pawel Maka and Thomas Vergutz and Siamak Mehrkanoon, 2021).

2.3.4. Daily maximum temperature forecasting in changing climate using a hybrid of Multidimensional Complementary Ensemble Empirical Mode Decomposition and Radial Basis Function Neural Network

En este trabajo (Lin, Tsai, y Chen, 2021) se desarrolla un nuevo modelo de red neuronal para detectar temperaturas extremas por medio de la predicción de la temperatura máxima del conjunto de los siguientes 7 días, manejando solamente el histórico de temperaturas. El conjunto de datos usado consta de 58 años (del 1/1/1960 a 31/12/2017, solamente de los meses entre julio y setiembre inclusive) de registros diarios de temperaturas máximas en 651 estaciones de medición de la ciudad de Taipéi. Del conjunto total de datos, los primeros 47 años (81 %) fueron usados para entrenamiento, y los siguientes 11 años (19 %) para testeo.

El objetivo en esta ocasión es desarrollar un modelo capaz de calcular la predicción de la temperatura máxima de los siguientes 7 días, teniendo la misma cantidad de registros en el pasado. El modelo usado en esta experimentación es el denominado: *MCEEMD + RBFNN*, el cual básicamente primero descompone todo el conjunto de datos en distintos subconjuntos que contemplan la multidimensionalidad de la información por medio de un modelo *MCEEMD* (*Multidimensional Complementary Ensemble Empirical Mode Decomposition*), y luego se le aplica una red neuronal *feed-forward*. La idea de esta red es des-

componer la entrada en distintas series de tiempo donde cada una es procesada por un *RBFNN* (*Radial Basis Function Neural Network*) y la salida de cada una es combinada para devolver el resultado. Cabe destacar que la arquitectura de este modelo no está basada en grafos.

Para medir y comparar el rendimiento de los distintos modelos se usaron las medidas *RMSE*, *MAPE* y el coeficiente de correlación (R). Esta última medida (cuya fórmula se ve planteada en la Ecuación 2.9), calcula el grado de correlación entre los valores predichos y los correspondientes valores esperados: con un rango entre -1 y 1, la correlación es más fuerte cuando la métrica se encuentra más próxima al segundo valor y cuanto más cerca al -1 la relación es inversamente proporcional.

$$\mathrm{R} = \frac{E[(Y_i - \mu_Y)(\hat{Y}_i - \mu_{\hat{Y}})]}{\sigma_Y \sigma_{\hat{Y}}}, \tag{2.9}$$

donde Y_i e $\hat{Y}_i$ son las variables aleatorias de los valores esperados y predichos respectivamente, calculando también la media (μ_Y y $\mu_{\hat{Y}}$) y la desviación estándar (σ_Y y $\sigma_{\hat{Y}}$) da cada una de estas.

Durante el proceso de experimentación se realizaron varias etapas de entrenamiento y testeo de modelos, variando la cantidad de nodos en cada capa. Esta etapa iterativa fue realizada manualmente, probando distintos valores a fuerza bruta, teniendo en cuenta los resultados óptimos al problema y el costo computacional de cada modelo (el costo incrementa si se agregan nodos).

	MAPE	**R**	**RMSE**
t	2.322727273	0.9281818182	0.948181818
t+1	2.785454545	0.8909090909	1.143181818
t+2	3.266818182	0.8522727273	1.317272727
t+3	3.281818182	0.8336363636	1.332727273
t+4	3.492727273	0.8277272727	1.411818182
t+5	3.837727273	0.7927272727	1.535909091
t+6	4.087272727	0.7640909091	1.627272727

Cuadro 2.3: Métricas del mejor modelo hallado para 7 días en el futuro.

El Cuadro 2.3 resume los resultados brindados en el artículo original. Se indican las métricas en promedio de todas las estaciones de medición del conjunto de los 7 días siguientes usando los criterios de evaluación mencionados, usando el mejor modelo hallado en el proceso de configuración de nodos por fuerza bruta. Allí se puede observar que este modelo tuvo buenos resultados, ya que se registró con un *RMSE* del día siguiente menor a 1 grado, menor a lo obtenido en trabajos anteriores.

Uno de los motivos por el que se pudo haber logrado dichos resultados radica en las características del conjunto de datos planteado. Debido a manejar una gran cantidad de estaciones de medición en un terreno relativamente chico (271.8 km^2, por lo tanto, no hay tanta variedad entre ellas) y que solo se

usa información de 3 meses consecutivos del año (julio, agosto y setiembre), se puede observar baja variabilidad de los datos. Esto es bien distinto en nuestro trabajo, donde manejamos mediciones en un territorio más extenso (aproximadamente 11 millones de km^2) en un período de tiempo mayor, lo que genera mayor variabilidad.

Otro punto interesante a tener en cuenta es el trabajo de configuración de las redes realizado. Donde evaluaron una secuencia manual de distintas configuraciones en lo que respecta a la cantidad de nodos de las redes. Este es un proceso usual en etapas de experimentación de modelos. En la Sección 3.2.3 se verá que en nuestro caso se hizo una tarea similar con los modelos planteados pero de forma automática, lo que permite no solo encontrar mejores configuraciones para los modelos, sino que también ahorrar tiempo de trabajo.

2.3.5. GraphCast: Learning skillful medium-range global weather forecasting

Los autores presentan *GraphCast* (Lam y cols., 2022), una técnica de aprendizaje automático basada en grafos para la predicción meteorológica de mediano plazo. La arquitectura del modelo se basa en el uso de una configuración del tipo "codificar-procesar-decodificar" basada de *GNN*. Los autores introducen una representación interna multi-grilla que permite lograr interacciones de largo alcance a través de la transmisión de pocos mensajes, logrando beneficiarse así de la habilidad de las *GNN* para el modelado de interacciones esparzas.

Para evaluar el desempeño de *GraphCast*, los autores lo comparan con dos modelos del estado del arte del momento. Por un lado, lo comparan con el modelo determinístico operacional de predicción meteorológica de mediano plazo más exacto hasta el momento, el *HRES* desarrollado por el *European Centre for medium-range forecasts (ECMWF)*[4]. Por otro lado, también lo comparan con una de las técnicas de aprendizaje automático pertenecientes al estado del arte del momento, *Pangu-Weather* (Bi, Kaifeng and Xie, Lingxiand Zhang, Henghengand Chen, Xinand Gu, Xiaotaoand Tian, Qi, 2023).

A la hora de realizar la comparación entre los modelos, los autores adoptan un enfoque similar al que usa el *ECMWF* para evaluar el desempeño de nuevos candidatos del *HRES* con sus versiones anteriores. Mediante el uso de un informe de resultados (*scorecard*), logran una comparación del desempeño de los modelos de predicción, teniendo en cuenta sus habilidades de manera integral sobre muchas variables clave. Usan el *RMSE (Root Mean Square Error)* y el *ACC (Anomaly Correlation Coefficient)*, los cuales miden la magnitud de las diferencias entre la predicción y el valor real, y qué tan bien el modelo pronostica las diferencias con la climatología respectivamente.

GraphCast fue entrenado con un subconjunto de 39 años proveniente del conjunto de datos *ERA5* (Hersbach y cols., 2020) del *ECMWF* en intervalos de 6 horas, en resolución de latitud-longitud horizontal de 0.25°, y 27 niveles

[4] https://www.ecmwf.int, accedido el 20/09/2023.

Tipo	**Nombre variable**	**Sigla**	**ECMWF ID de Parámetro**	**Rol (período de acumulación, si corresponde)**
Atmosférica	Geopotencial	z	129	Entrada/Predicho
Atmosférica	Humedad específica	q	133	Entrada/Predicho
Atmosférica	Temperatura	t	130	Entrada/Predicho
Atmosférica	Componente U del viento	u	131	Entrada/Predicho
Atmosférica	Componente V del viento	v	132	Entrada/Predicho
Atmosférica	Velocidad vertical	w	135	Entrada/Predicho
Única	Temperatura a 2 metros	2t	167	Entrada/Predicho
Única	Componente U del viento a 10 metros	10u	165	Entrada/Predicho
Única	Componente V del viento a 10 metros	10v	166	Entrada/Predicho
Única	Presión media al nivel del mar	msl	151	Entrada/Predicho
Única	Precipitaciones totales	tp	228	Entrada/Predicho (6h)
Única	Radiación solar incidente TOA	tisr	212	Entrada (1h)
Estática	Geopotencial en superficie	z	129	Entrada
Estática	Máscara tierra-mar	lsm	172	Entrada
Estática	Latitud	n/a	n/a	Entrada
Estática	Longitud	n/a	n/a	Entrada
Reloj	Hora local del día	n/a	n/a	Entrada
Reloj	Progreso del año transcurrido	n/a	n/a	Entrada

Cuadro 2.4: Variables usadas en el *dataset* proveniente de *ECMWF*. La columna "*Tipo*" indica si la variable representa una propiedad estática, una propiedad de un único nivel que varía en el tiempo, o una propiedad atmosférica que varía en el tiempo. La columna "Sigla" hace referencia a la etiqueta de la variable para el ECMWF's. La columna "ECMWF ID de Parámetro" hace referencia a la etiqueta numérica del ECMWF's y puede ser usada para construir la URL para la descripción de la variable agregándolo como sufijo al siguiente prefijo, reemplazando "ID" con el código numérico: *https://apps.ecmwf.int/codes/grib/param-db/?id=ID*. La columna "Rol" indica si la variable es algo que *GraphCast* recibe como entrada y además predice, o si solamente la usa como entrada. Fuente: (Lam y cols., 2022).

de presión atmosférica vertical. En el Cuadro 2.4 se pueden ver detalladas las variables utilizadas.

GraphCast logra hacer una predicción de 10 días en intervalos de 6 horas de 5 variables de superficies y 6 variables atmosféricas, cada una en 37 niveles de presión verticales en una grilla de latitud-longitud de 0.25°, que corresponde a una resolución de aproximadamente 25 × 25 kilómetros en el ecuador. Alcanza a ser más preciso que el *HRES* (el sistema operacional determinístico del *ECMWF*) en un 90.0 % de las 2760 variables e intervalos de tiempo en los que se evaluaron. Una de las razones principales por las cuales *GraphCast* logra desempeñarse mejor que *HRES* es porque fue entrenado usando datos directos y en principio, logra captar fenómenos meteorológicos, como masas de aire, frentes, tormentas, etc., a escalas que no están representadas explícitamente dentro de los sistemas basados en *NWP (Numerical Weather Prediction)*. Por otra parte, *GraphCast* logra superar el desempeño de *Pangu-Weather* en un 99.2 % de los 252 objetivos reportados. En el Cuadro 2.5 se detallan que variables meteorológicas fueron las

Nombre variable	**Abreviación**
Geopotencial a 5.5 km	z500
Temperatura a 5.5 km	t500
Temperatura a 1.4 km	t850
Humedad específica a 5.5 km	q500
Componente U del viento a 5.5 km	u500
Componente V del viento a 5.5 km	v500
Temperatura a 2 metros	2t
Componente U del viento a 10 metros	10u
Componente U del viento a 10 metros	10v

Cuadro 2.5: Variables usadas para la evaluación del desempeño entre los modelos *GraphCast* y *Pangu-Weather*. Se utilizaron un total de 9 variables, haciéndose predicciones de 7 días hacia adelante en intervalos de 6hs.

utilizadas para la comparación entre los modelos.

2.3.6. Cuadro comparativo entre modelos estudiados.

El Cuadro 2.6 resume algunos datos de importancia de los modelos presentados anteriormente, remarcando diferencias con nuestro trabajo.

Es importante destacar que no sería adecuado comparar los distintos modelos por medio de este cuadro, ya que si bien todos buscan predecir temperaturas, cada trabajo fue realizado bajo distintas condiciones. Entre estas diferencias se destacan el origen y tamaño del conjunto de datos, el tamaño de los pasos de entrada y el de salida.

	Forecasting The Air Temperature at a Weather Station Using Deep Neural Networks	**Purely data-driven medium-range weather**	**TENT**	**MCEEMD + RBFNN**	**GraphCast**
Modelo usado	MLP, LSTM, CNN + LSTM	Fully connected CNNs. directos y continuos	TENT, basado en transformers	MCEEMD + RBFNN	*GraphCast*
Es basado en grafos	No	No	No	No	Si
Unidad de medición	Días	Horas	Días y Horas	Días	Horas
Datos de entrada	Velocidad del viento, precipitaciones, nevadas, temperaturas máxima, media y mínima	Temperaturas, viento, humedad, precipitaciones	Temperaturas, precipitaciones, velocidad del viento, visibilidad, presión, humedad	Temperaturas	Variables especificadas en el Cuadro 2.4
Cantidad de estaciones	1	2048	19 / 30	651	Grilla de latitud/longitud con resolución de 0.25° e incrementos de 1 hora.
Período	10 años	40 años	15 años / 5 años	58 años	39 años
Datos a predecir	Temperatura promedio del día siguiente. Temperatura promedio del conjunto de los 10 días	Información atmosférica, temperatura, precipitaciones.	Temperaturas	Temperatura	*HRES*: 2.4 *Pangu-Weather*: 2.5
Medida de evaluación	*RMSE* + *MAPE*	*RMSE* ponderado por ubicación	*MAE*, *MSE*	*MAPE*, *R*, *RMSE*	*RMSE* normalizado, *ACC* normalizado
Modelo Baseline	No	IFS	Transformer, 3d CNN, LSTM, ConvLSTM	No	*HRES*, *Pangu-Weather*
Tamaño dataset	train: 80 %, 2920 registros test: 20 %, 730 registros	train: 95 %, 332880 registros test: 5 %, 17520 registros	Estados Unidos y Canada train: 80 %, 35040 registros test: 20 %, 8760 registros Europa train: 80 %, 4380 registros test: 20 %, 1095 registros	train: 80 %, 5828 registros test: 20 %, 1351 registros	train: 97 %, 38 años test: 3 %, 1 año
RMSE	MLP: 0.2 °F LSTM: 0.1 °F CNN+LSTM: 0.075 °F	modelo directo: $\sim$ 0,9 °K	MAE Europa: 1.84 °K (predicción dentro de dos días)	0.94 °C	$\sim$ 0,5 °K

Cuadro 2.6: Tabla comparativa entre modelos estudiados.

2.4. Graph WaveNet

2.4.1. ¿Qué busca resolver?

Graph WaveNet(Wu y cols., 2019) es una arquitectura de aprendizaje profundo basada en grafos, utilizada para resolver problemas de predicción con datos espacio-temporales. Los autores la presentan como una arquitectura que hace un mejor uso de los datos espaciales del problema en comparación a otras similares. Logra resolver dos problemas comunes en este tipo de arquitecturas:

- capturar dependencias espaciales escondidas en los datos; y
- capturar tendencias temporales en secuencias largas de datos.

Los autores de *Graph WaveNet* implementaron una versión de la arquitectura en Python utilizando *pytorch*[5], la cual se encuentra disponible en GitHub[6]. A continuación describiremos la arquitectura basándonos en (Beyer, Ahmad, Yang, y Rodríguez-Bocca, 2023).

2.4.2. ¿Cómo lo resuelve? (arquitectura)

Graph WaveNet se destaca principalmente por introducir dos características que buscan resolver los problemas planteados en la sección anterior:

- implementa una matriz de adyacencias auto-adaptativa, mediante la que se pueden aprender dependencias espaciales escondidas entre los nodos del grafo; y
- presenta una arquitectura que logra capturar dependencias espacio-temporales de forma eficiente y efectiva.

La arquitectura de *Graph WaveNet* es la que se muestra en la Figura 2.6. La misma está conformada por una pila de capas espacio-temporales y una capa de salida.

Por un lado, su capa de salida es una combinación de una función de activación *ReLu* y transformaciones lineares necesarias para obtener la salida de las dimensiones correspondientes.

Por otro lado, cada capa espacio-temporal está compuesta por una celda GCN (*Graph Convolution Layer*) y una celda G-TCN (*Gated Temporal Convolution Layer*). Estos conceptos se desarrollarán a continuación. Al apilarse múltiples capas espacio-temporales, la arquitectura es capaz de manejar dependencias temporales a distintas escalas. Por ejemplo, la capa inferior recibe información temporal a corto plazo, mientras que en la capa superior maneja información temporal a largo plazo.

[5] https://pytorch.org/, accedido el 20-03-2023
[6] https://github.com/nnzhan/Graph-WaveNet, accedido el 20-03-2023

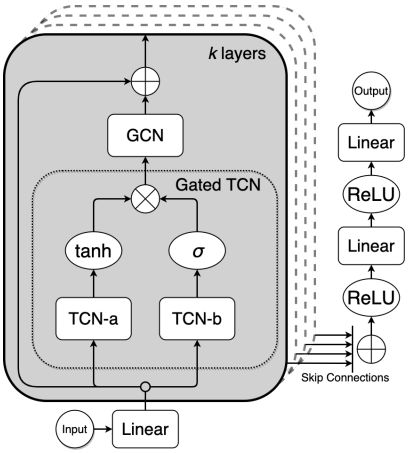

Figura 2.6: Arquitectura *Graph WaveNet* conformada por una serie de k capas espacio-temporales (G-TCN más GCN) y unas capas lineales finales (a la derecha). Fuente (Beyer y cols., 2023).

Graph Convolution Layer (GCN)

Para la extracción de la información espacial, *Graph WaveNet* hace uso de una capa convolucional para grafos o *GCN* (*Graph Convolutional Network*) (Wu y cols., 2020). Las *GCN* pueden verse como bloques que aprenden usando datos que siguen una estructura de grafos, permitiéndonos así trabajar con datos no Euclidianos. Se suelen usar para problemas de clasificación de nodos, clasificación de grafos y predicción de conexiones entre nodos, entre otros. A continuación formalizamos el funcionamiento de la arquitectura *GCN* basado en el desarrollo del *paper* original.

Kipf (Kipf y Welling, 2017) propuso una primera aproximación del filtro espectral de Chebyshev, con el que se logra suavizar la señal del nodo combinando y transformando la información de sus nodos vecinos. Una de sus mayores ventajas es el soporte a señales multidimensionales. Sea $\tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$ la matriz de adyacencia normalizada con lazos, $\mathbf{X} \in \mathbb{R}^{N \times D}$ las señales de entrada, $\mathbf{Z} \in \mathbb{R}^{N \times M}$ la salida, y $\mathbf{W} \in \mathbb{R}^{D \times M}$ la matriz de parámetros del modelo. Entonces la capa convolucional queda definida por:

$$\mathbf{Z} = \tilde{\mathbf{A}}\mathbf{X}\mathbf{W}. \tag{2.10}$$

Se utiliza la matriz de adyacencia normalizada para evitar que las señales se amplifiquen al pasar por la capa, y se agregan lazos para hacer explícita la dependencia temporal de la señal sobre el mismo nodo.

Por otro lado, Li (Li, Yu, Shahabi, y Liu, 2018) propuso una capa convolucional con difusión, la cual es efectiva para el modelado espacio-temporal. Modela

el proceso de difusión de las señales sobre los nodos del grafo en K pasos finitos:

$$\mathbf{Z} = \sum_{k=0}^{K} \mathbf{P}^k \mathbf{X} \mathbf{W}_k, \tag{2.11}$$

donde $\mathbf{P}^k$ representa la serie de potencias de la matriz de transición. Para el caso de un grafo no-direccionado, $\mathbf{P} = \mathbf{A}/rowsum(\mathbf{A})$. En el caso de un grafo direccionado, el proceso de difusión puede tener dos direcciones: *forward* (hacia delante) y *backward* (hacia atrás). La matriz de transición *forward* puede definirse como $\mathbf{P}_f = \mathbf{A}/rowsum(\mathbf{A})$, mientras que la matriz de transición *backward* se define como $\mathbf{P}_b = \mathbf{A}^T/rowsum(\mathbf{A}^T)$. Con estas matrices de transición, la capa de convolución con difusión puede ser escrita como:

$$\mathbf{Z} = \sum_{k=0}^{K} \mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k2}. \tag{2.12}$$

Los autores de *Graph WaveNet* proponen el uso de una matriz de adyacencia auto-adaptativa $\tilde{\mathbf{A}}_{adp}$. Esta matriz no necesita ser inicializada con información del grafo y se aprende mediante descenso por gradiente estocástico, permitiéndole así descubrir posibles dependencias ocultas entre nodos. La matriz de adyacencia auto-adaptativa propuesta puede definirse como:

$$\tilde{\mathbf{A}}_{adp} = SoftMax(ReLU(\mathbf{E}_1 \mathbf{E}_2^T)), \tag{2.13}$$

donde $\mathbf{E}_1, \mathbf{E}_2 \in R^{N \times c}$ son dos diccionarios de *node embeddings* inicializados aleatoriamente, los cuales representan los nodos de origen y los nodos objetivo, respectivamente. Al multiplicarse $\mathbf{E}_1$ y $\mathbf{E}_2$ se obtienen las dependencias espaciales entre los nodos de origen y objetivo. Se usa la función de activación $ReLU$ para eliminar las conexiones débiles y la función $SoftMax$ para normalizar la matriz. Por tanto, para el caso en el que la información inicial del grafo no se encuentra disponible, los autores proponen entonces la ecuación:

$$\mathbf{Z} = \sum_{k=0}^{K} \tilde{\mathbf{A}}_{adp}^k \mathbf{X} \mathbf{W}_k. \tag{2.14}$$

Es posible combinar ambas salidas (difusión y adaptativa). Por ejemplo, cuando conocemos una estructura del grafo dominante en la interrelación espacial del problema (por ejemplo, las distancias entre los nodos), pero además queremos mejorar dicha interrelación mediante la adaptación. Entonces, podemos usar la fórmula general de la capa convolucional para grafos (GCN):

$$\mathbf{Z} = \sum_{k=0}^{K} \left(\mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k2} + \tilde{\mathbf{A}}_{adp}^k \mathbf{X} \mathbf{W}_{k3} \right), \tag{2.15}$$

esta fórmula genérica es la implementada en la arquitectura *Graph WaveNet*.

Gated Temporal Convolution Layer (TCN)

Para el aprendizaje de dependencias temporales en grafos, se suele hacer uso principalmente de dos enfoques de redes neuronales, las redes basadas en *CNN* (*Convolutional Neural Network*) o redes basadas en *RNN* (*Recurrent Neural Network*).

Graph WaveNet implementa un enfoque *CNN* y lo adopta a la convolución causal dilatada (*Dilated Causal Convolutions*) (Yu y Koltun, 2016) como su celda temporal (*TCN*). Este tipo de convolución es capaz de manejar secuencias de largo alcance de una manera no-recursiva, lo que permite paralelizar cómputo y resuelve otro tipo de problemas de performance que tienen las redes *RNN*, como es la explosión/desaparición del gradiente.

Las convoluciones con causalidad dilatada son un caso especial de las convoluciones estándares de dimensión 1D, donde recorren las entradas salteándose entradas cada cierta distancia. Esto puede verse en la Figura 2.7.

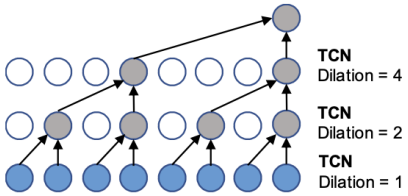

Figura 2.7: Convolución casual dilatada con kernel tamaño 2 y factor de dilatación k. Extraído de (Wu y cols., 2019)

Matemáticamente, pueden ser representadas de la siguiente forma: sea una secuencia $1D$ de entrada, $\mathbf{x} \in \mathbb{R}^T$, y un filtro, $\mathbf{f} \in \mathbb{R}^k$, entonces, la convolución causal dilatada de $\mathbf{x}$ con $\mathbf{f}$ en el paso t se encuentra representada como:

$$\mathbf{x} \star \mathbf{f}(t) = \sum_{s=0}^{K-1} \mathbf{f}(s)\mathbf{x}(t - d \times s), \tag{2.16}$$

donde d es el factor de dilatación que determina la distancia de cuántos valores saltearse.

Cuando se apilan una serie de convoluciones causales dilatadas cuyos factores de dilatación son crecientes, se logra que el campo receptivo del modelo crezca exponencialmente. Lo que le permite capturar secuencias más largas de datos con menor cantidad de capas.

Graph WaveNet utiliza un mecanismo de compuerta (*Gating Mechanism*) en la salida de su celda temporal G-TCN (*Gated TCN*), la cual consiste en dos capas temporales paralelas (TCN-a y TCN-b) que solo se diferencian en la función de activación (TCN-a usa la tangente hiperbólica, y TCN-b usa la

sigmoide). Esta puede definirse como:

$$\mathbf{Z} = \tanh(\mathbf{W}_1 \star \mathbf{X} + b_1) \odot \sigma(\mathbf{W}_2 \star \mathbf{X} + b_2), \tag{2.17}$$

donde $\mathbf{X} \in \mathbb{R}^{N \times D \times T}$ son las señales de entrada, $\mathbf{Z}$ es la salida, $\mathbf{W}_1$,$\mathbf{W}_2$, b_1 y b_2 los parámetros del modelo, $\odot$ es el producto punto, y $\star$ es la convolución causal dilatada (de parámetro d).

Los autores eligieron el error absoluto medio o *MAE* (*Mean Absolute Error*) como la función objetivo de *Graph WaveNet*. El mismo queda definido con la Ecuación (2.18).

$$\mathcal{L}(\hat{\mathbf{X}}^{(t+1):(t+T)}; \mathbf{\Theta}) = \frac{1}{TND} \sum_{i=1}^{i=T} \sum_{j=1}^{j=N} \sum_{k=1}^{k=D} |\hat{\mathbf{X}}_{jk}^{(t+i)} - \mathbf{X}_{jk}^{(t+i)}|, \tag{2.18}$$

donde N es el número de nodos, C son las dimensiones escondidas y T es el largo de secuencia, y corresponden a los tensores de 3 dimensiones $[N, C, T]$ de entradas de las celdas *GCN*.

2.4.3. Ventajas y desventajas en el uso de Graph WaveNet

Graph WaveNet es una arquitectura novedosa para la resolución de problemas de grafos. En función de lo detallado por los autores, podemos identificar en su uso las siguientes ventajas:

- incluye manejo de información en forma de grafos por defecto;
- logra capturar dependencias espaciales escondidas entre nodos;
- tiene buen desempeño al trabajar con muchas características (*features*); y
- es rápido en su entrenamiento.

Por otro lado, el uso de esta arquitectura presenta algunas desventajas:

- al tratarse de una arquitectura novedosa existe poca documentación e investigación sobre la misma; y
- requiere uso de GPU que es un recurso costoso.

2.5. XGBoost

2.5.1. ¿Qué busca resolver?

XGBoost[7] [8] (*eXtreme Gradient Boosting*) (Chen y Guestrin, 2016) se presenta como una librería que permite usar modelos de *machine learning* para

[7] XGBoost Documentation: https://xgboost.readthedocs.io/en/stable/index.html, accedido el 20-03-2023

[8] Scalable and Flexible Gradient Boosting: https://xgboost.ai, accedido el 20-03-2023

resolver problemas de predicciones tanto de clasificación como de regresión. En particular, implementa modelos de *gradient boosting trees* con optimizaciones que resultan en eficiencia, flexibilidad y portabilidad.

Es una librería de código abierto, disponible en múltiples lenguajes de programación (C, C++, Java, Python, R, Ruby, Swift, Julia, Perl, Scala) y además se puede ejecutar en diferentes sistemas operativos (Linux, Windows y Mac OS).

XGBoost se presenta como un modelo que corre más rápido en comparación a sus similares. Esto se debe a las decisiones de optimización en la implementación agregadas al modelo de *gradient boosting* existente, y también a la posibilidad de ejecutarse de forma distribuida.

2.5.2. ¿Cómo lo resuelve? (arquitectura)

Tal como se mencionaba anteriormente, *XGBoost* es un sistema el cual **optimiza el entrenamiento y ejecución** de un modelo basado en *boosting trees* (acoplamiento de árboles de decisión).

Árbol de decisión.

Un árbol de decisión es un método de aprendizaje supervisado. En el proceso de entrenamiento se cuenta con un conjunto de datos de entrada, x_i, y un conjunto de datos de salida esperada, y_i (resultado esperado de la clasificación o regresión). Tiene como propósito la generación de un modelo capaz de la predicción de un valor objetivo, partiendo de ciertos datos de entrada (características). Para esto define un conjunto de reglas, las cuales son las encargadas de tomar una secuencia de decisiones y finalmente concluir un resultado. Lleva el nombre de "árbol", ya que el conjunto de todas las posibles decisiones (y resultados) se puede modelar de esta forma. Cada una de las secuencias de decisiones forma un camino desde el conjunto de entrada hasta uno de los valores resultantes, cada regla se la reconoce como un nodo interno al árbol y puede tomar dos opciones de salida. Las hojas del árbol generado son las distintas conclusiones que el modelo puede tomar. Durante el proceso de construcción, los criterios de decisión se determinan de forma tal que se minimice la incertidumbre (o entropía) en la información, teniendo en cuenta el rango de valores de parámetro en el conjunto de datos de entrada. Por ejemplo, en la Figura 2.8, extraída de la publicación que presentaron los autores del modelo (Chen y Guestrin, 2016), se puede observar un ejemplo reducido de árbol de decisión en donde se agrupan a las personas por su interés en los juegos de computadora con un único criterio: el límite de edad. Para cada caso, se chequea la característica de edad y se aplica el criterio de decisión, el nodo final depende de si la respuesta es verdadera o falsa.

Boosting trees.

La técnica de *boosting* busca combinar un conjunto de algoritmos "débiles" de modo de lograr uno más complejo y robusto. En este caso, se busca la conca-

tenación de un conjunto de árboles de decisión para obtener un modelo final que ofrezca mejores resultados que cada uno individualmente. Para la construcción de cada árbol, se basa en los árboles creados anteriormente buscando minimizar el error aplicando descenso por gradiente. Un ejemplo de esto se puede ver en la Figura 2.9, donde se sigue con el mismo objetivo que con el árbol de la Figura 2.8 (interés por los juegos de computadoras), pero para obtener mejores resultados se usa la información agrupada de dos árboles de decisión.

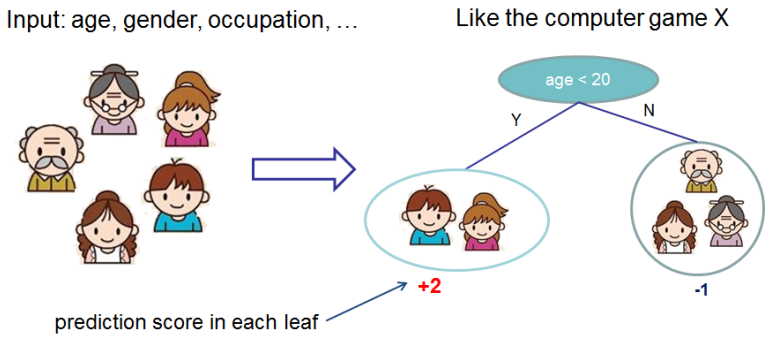

Figura 2.8: Representación gráfica de un árbol de decisión. Fuente: (Chen y Guestrin, 2016).

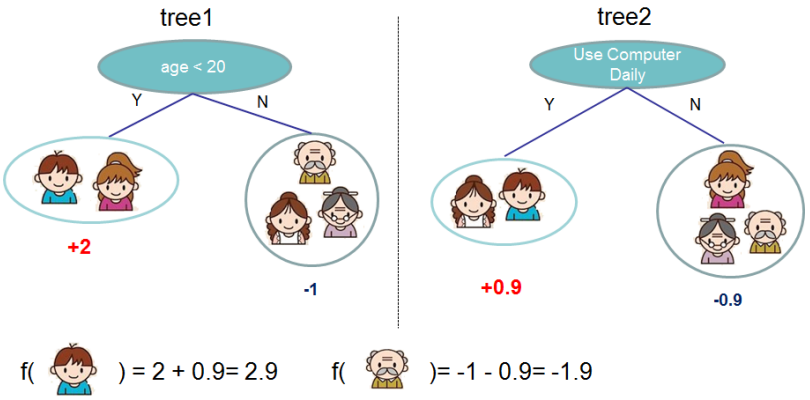

Figura 2.9: Ejemplo de ejecución de *boosting trees*: la clasificación final para cada individuo depende de los resultados de cada árbol de decisión. Fuente: (Chen y Guestrin, 2016).

A continuación formalizamos el método de *XGBoost*, siguiendo la referencia (Chen y Guestrin, 2016). Sea F el conjunto de todos los posibles árboles a construir basándose en el dominio de las características de entrada. Desarrollada en la Ecuación (2.19), se puede ver como cada árbol es manejado como

una función $f(x)$, reflejando el resultado de la predicción para un conjunto de entrada x, la cual se basa en hallar el peso (hoja en el árbol) correspondiente a x.

$$\mathrm{F} = \{f(x) = w_{q(x)}\}, \tag{2.19}$$

Donde w ($w \in \mathbb{R}^T$) es el vector con los pesos de cada hoja del árbol, q ($q : \mathbb{R}^m \to T$) refiere al camino en el mismo tomado para llegar hasta la hoja correspondiente, m es la cantidad de características de entrada, T indica la cantidad de hojas en el árbol (es decir, los nodos finales).

La predicción final resulta en la suma de todos los árboles del conjunto. Por lo tanto:

$$\widehat{y} = \sum_{k=1}^{K} f_k(x), f_k \in F,$$

siendo K el número de árboles generados.

La creación de nuevos árboles se hace en base de los anteriores. Por lo tanto, al momento de generar un nuevo árbol de decisión f_t, se tiene que:

$$\widehat{y_i}^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \widehat{y_i}^{(t-1)} + f_t(x_i).$$

Y los pesos de las hojas del nuevo árbol se calculan al buscar minimizar la diferencia con el verdadero valor de y. Para esto, minimizará la función objetivo:

$$\begin{aligned} obj^{(t)} &= \textstyle\sum_{i=1}^{n} \mathcal{L}(y_i, \widehat{y_i}^{(t)}) + \sum_{i=1}^{t} \omega(f_i), \\ &= \textstyle\sum_{i=1}^{n} \mathcal{L}(y_i, \widehat{y_i}^{(t-1)} + f_t(x_i)) + \omega(f_t) + cte, \end{aligned}$$

siendo $\mathcal{L}(y_i, \widehat{y_i}^{(t)})$ la función de pérdida entre el valor y real y el $\widehat{y_i}^{(t)}$ predicho.

Para hallar los pesos del nuevo árbol, se aplica el desarrollo de Taylor de segundo orden basándose en la función de pérdida, y los despeja en función de las derivadas.

2.5.3. ¿Qué características agrega *XGBoost*?

Existen varias librerías basadas en *gradient boosting trees*, como por ejemplo CatBoost (Dorogush, Ershov, y Gulin, 2018) y LightGBM (Ke y cols., 2017). *XGBoost* agrega las siguientes características de optimización a la técnica básica explicada en los puntos anteriores:

- ***Weighted quantile for aproximated tree learning.***
  Para crear cada árbol, cada nodo de decisión debería considerar todos los parámetros de entrada y para cada uno todos los posibles valores de criterio dentro del rango de estos para identificar cuál divide mejor la información (minimizando la incertidumbre). En lugar de esto, *XGBoost* agrupa los posibles valores en cuantiles por peso (no agrupa por igual

cantidad, sino de forma tal que el valor final acumulado para cada grupo sea el mismo) y toma un representante de cada cuantil, obteniendo entre ellos el óptimo.

A su vez, tiene la capacidad de no seleccionar todas las características y todos los registros en cada iteración.

- ***Parallel learning.***
  Para optimizar la selección, subdivide todo el conjunto de datos en grupos más pequeños, repartiendo los mismos entre distintos hilos o máquinas para acelerar el proceso.

- ***Sparcity aware algorithm.***
  Tiene la capacidad de manejar datos nulos, teniendo una decisión "por defecto" dentro del árbol para esos casos.

- ***Cache aware access.***
  Almacena en el caché los datos usados con frecuencia (gradientes y hessianos) para calcular los pesos, obteniendo el acceso más rápido posible.

- ***Block of out of core computation.***
  En casos de grandes bloques de información, donde no puede almacenarse todo en caché y RAM, se debería almacenar el resto en el disco, lo que genera accesos lentos. Para evitar esto se comprime el conjunto de datos, bajo el supuesto que las tareas de compresión y descompresión son más rápidas que el acceso y lectura a disco.

  También tiene la capacidad de separar el dataset en distintos discos (si es que se tiene más de uno) para poder paralelizar la lectura y escritura.

- ***Tree Pruning.***
  El algoritmo cuenta con un criterio de parada en la construcción de árboles. Uno de sus parámetros, `max_depth`, permite indicar la altura máxima de los árboles de decisión, evitando subdividir en conjuntos muy pequeños, lo que provoca un sobre ajuste (*overfitting*) en el modelo basándose en los datos de entrada. A su vez, durante la construcción, se calcula un parámetro de calidad en las divisiones del árbol, el cual también se tiene en cuenta para cortar ramas del mismo: si el valor calculado en un nodo no supera cierto registro (`gama`, también configurable), la rama es eliminada. El hecho de cortar y poder reducir el tamaño de los árboles presenta una mejora en lo que respecta a la performance, ya que se hacen menos cálculos.

XGBoost ofrece un conjunto de parámetros configurables que se dividen en parámetros generales, de *boosting* (relacionados con el comportamiento durante el entrenamiento y la concatenación de árboles) y finalmente parámetros de la tarea de aprendizaje. Adicionalmente, al conjunto de parámetros y a las características de optimización del cómputo mencionados anteriormente, la herramienta ofrece funcionalidades que ayudan durante el entrenamiento y la comprensión del modelo creado, las cuales se comentarán en la Sección 3.2.5, donde

explicaremos en detalle sus hiper-parámetros. Algunas funcionalidades son: uso de GPU, manejo de variables categóricas (variables que manejan un dominio discreto), posibilidad de crear modelos con más de una salida y el análisis de la importancia de los parámetros de entrada.

2.5.4. Ventajas y desventajas en el uso de *XGBoost*

Al momento de su publicación, se reporta[9] que *XGBoost* fue usado para varias soluciones ganadoras dentro de las competencias disponibles en el sitio de Kaggle [10]. En el 2015, de las 29 soluciones ganadoras, 17 implementaban esta librería. También se registró su uso en propuestas ganadoras del KDDCup[11] 2015.

En (Chen y Guestrin, 2016) se indica que se ha utilizado esta herramienta como solución para problemas de: predicción en ventas, clasificación de eventos físicos, de consumo de energía, clasificación de texto en la Web, predicción del comportamiento de clientes, detección de movimiento, predicción de tasa de clics en publicidades, clasificación de *malware*, clasificación de productos, predicción de posibilidad de riesgo y predicción de tasa de abandono masivo en cursos en línea.

De lo descrito anteriormente se pueden destacar las siguientes **ventajas**:

- multilenguaje: está disponible en varios lenguajes de programación, incluso soportado en software modernos;
- algoritmo paralelizable: permite la ejecución de procesos en forma paralela, optimizando el uso de recursos;
- fácil escalabilidad;
- soporta cálculos basándose en distintas funciones de pérdida;
- algoritmo rápido: gracias a su implementación, presenta mejores tiempos de respuesta, tanto en entrenamiento como en predicción respecto a modelos similares; y
- en general ofrece una buena performance sin mucho esfuerzo de configuración.

Según distintos foros en donde se ha discutido el uso del método[12] [13] [14], se identifican las siguientes desventajas:

[9] https://github.com/dmlc/xgboost/blob/master/demo/README.md#machine-learning-challenge-winning-solutions, accedido el 20-03-2023

[10] Kaggle, *Allstate Claim Prediction Challenge*: https://www.kaggle.com/c/ClaimPredictionChallenge, accedido el 20-03-2023

[11] KDD Cup 2015: http://moocdata.cn/challenges/kdd-cup-2015, accedido el 20-03-2023

[12] Krayonnz-What are the advantages and disadvantages of XGBoost?: www.krayonnz.com, accedido el 20-03-2023

[13] Quora-What are the advantages and disadvantages of XGBoost?: https://www.quora.com/, accedido el 20-03-2023

[14] Introduction to boosted decision trees: https://indico.fnal.gov/event/15356/contributions/31377/attachments/19671/24560/DecisionTrees.pdf, accedido el 20-03-2023

- las variables de entrada deben ser numéricas;
- tiende a sobre-entrenar si no se controla correctamente;
- requiere precisión al momento de trabajar con la configuración de hiperparámetros; y
- puede suceder que no sea performante cuando la entrada se presenta dispersa o sin una estructura correctamente definida.

XGBoost se ha utilizado exitosamente para resolver problemas predictivos con datos interrelacionados espacio-temporales (Nyéki, A. and Kerepesi, C. and Daróczy, B. and Benczúr, A. and Milics, G. and Nagy, J. and Harsányi, E. and Kovács, A. J. and Neményi, M., 2021) (Sun, Sun, y Jiao, 2021) (Schönauer y cols., 2022). En estos casos, es imprescindible realizar una profunda ingeniería de características (*feature engineering*) para extraer atributos relevantes al problema y hacer explícitas las interdependencias. Debido a que la arquitectura no es capaz de realizar esta tarea automáticamente durante el proceso de entrenamiento, uno tiene que llevarla adelante previo a ejecutar esta etapa. Sin ella, no se obtienen buenos resultados. Esto fue realizado y constatado en el marco de nuestro proyecto.

Capítulo 3

Datos y Metodología

En esta Sección, nos enfocaremos en dos actividades críticas para el desarrollo de nuestro proyecto de investigación: el análisis y procesamiento de los datos, así como la metodología que guió la creación de nuestros modelos.

En la Sección 3.1 presentamos el análisis de datos, que desempeña un papel fundamental al permitirnos comprender la naturaleza de la información recopilada. Además, nos ayuda a identificar patrones y casos específicos que pueden resultar valiosos al entrenar nuestros modelos. A continuación, abordaremos la etapa de procesamiento de datos, que incluye tareas como la limpieza, corrección y transformación de los datos, con el propósito de prepararlos para su uso en nuestras investigaciones.

En la Sección 3.2 presentamos la metodología de trabajo, componente central en este proceso, ya que establece las características y decisiones que orientaron nuestro proyecto para cumplir con el objetivo establecido. A pesar de que cada uno de los modelos planteados tiene sus particularidades inherentes, buscamos asegurar condiciones equitativas a la hora de construir y evaluar estos modelos. Este punto es de suma importancia, ya que nos proporciona las bases para realizar una comparación rigurosa y justa entre las distintas técnicas que hemos empleado.

3.1. Datos utilizados en este trabajo

A continuación se desarrolla sobre los datos con los que trabajamos: ¿dónde fueron obtenidos?, ¿qué procesamientos y transformaciones se le hicieron?

3.1.1. Origen y armado del dataset

En el proceso de creación de nuestro conjunto de datos, inicialmente recopilamos información de dos fuentes principales: el Instituto Uruguayo de Meteorología (INUMET, por medio de solicitud directa) y consultando la página del Instituto Nacional de Meteorología de Brasil. Sin embargo, nos encontramos

con varios desafíos en los datos recopilados, que incluían incongruencias, gran cantidad de datos faltantes, variabilidad y errores. A pesar de nuestros esfuerzos iniciales en la limpieza y procesamiento de los datos, no logramos obtener resultados satisfactorios. Para lograr mejorar la calidad de nuestros datos, solicitamos ayuda al equipo de trabajo de *ClimateDL*[1], que ya disponía de una fuente de datos depurada y una metodología sólida. *ClimateDL* se encuentra enmarcado en *CLIMAT-AmSud*[2], un programa de carácter científico que busca resolver problemas climatológicos con un equipo interdisciplinario e internacional. De esta forma compartimos nuestros datos con ellos y les solicitamos aplicar la misma metodología que utilizaban en sus propios datos. Nuestros datos, en conjunto con los de ellos, pasaron por un control de calidad realizado en el lenguaje *R*[3] con el paquete *climdex*[4]. En total contaban con 206 estaciones a las cuales se les corrigieron datos anormales y errores, seleccionando aquellas que no superaran cierto porcentaje de datos faltantes. Como resultado de esta colaboración obtuvimos un conjunto de datos analizados y procesados por expertos, enriquecido con datos complementarios proporcionados por *ClimateDL*, lo que mejoró significativamente la calidad y utilidad de nuestro conjunto de datos.

El conjunto final incluye información de temperaturas diarias mínimas, máximas y precipitaciones de 137 estaciones de medición distribuidas en parte del territorio sudamericano. Las temperaturas se midieron con termómetro (registradas en °C) y las precipitaciones con pluviómetro (registradas en mm). Estas mediciones fueron provistas por distintas instituciones, de distintos países:

- Servicio Meteorológico Nacional de Argentina[5] (Argentina).
- Dirección de Meteorología e Hidrología[6] (Paraguay).
- Instituto Uruguayo de Meteorología[7] (Uruguay).
- Instituto Nacional de Meteorología[8] (Brasil).
- Centro de Investigaciones sobre Clima y Resiliencia de Chile[9] (Chile).

3.1.2. Formato

El conjunto de datos de origen se encuentra principalmente dividido en 3 archivos, los cuales contienen los registros de temperaturas máximas, temperaturas mínimas y las precipitaciones en todas las estaciones manejadas. Además, contamos con un cuarto archivo adicional que contiene referencias sobre las

[1] https://climatedl.pages.fing.edu.uy/website/about.html, accedido el 02/08/2023
[2] https://www.sticmathamsud.org/sitio/climat/presentacion/, accedido el 02/08/2023
[3] https://www.r-project.org/, accedido el 20/9/2023
[4] https://www.rdocumentation.org/packages/ClimProjDiags/versions/0.2.1/topics/Climdex, accedido el 20/9/2023
[5] https://www.smn.gob.ar, accedido el 02/04/2023
[6] https://www.meteorologia.gov.py, accedido el 02/04/2023
[7] https://www.inumet.gub.uy, accedido el 02/04/2023
[8] https://portal.inmet.gov.br, accedido el 02/04/2023
[9] https://www.cr2.cl, accedido el 02/04/2023

estaciones de medición: aportando principalmente el identificador y ubicación geográfica (latitud y longitud) de las mismas.

Los archivos con las mediciones están en forma tabular: las columnas registran las estaciones y las filas indican las mediciones correspondientes a cada fecha para todas las estaciones. El Cuadro 3.1 es un ejemplo de este formato, para un subconjunto de los datos. El conjunto de datos de origen tiene datos faltantes, los cuales se encuentran representados mediante el número *-99.9*.

YYYY/MM/DD	11000	11001	11002	11005	11006	11007	11008
1977/01/01	31.6	23.0	-99.9	-99.9	30.6	28.8	31.0
1977/01/02	31.5	28.7	-99.9	-99.9	30.4	27.0	32.2
1977/01/03	25.4	25.4	-99.9	-99.9	30.2	29.0	28.8
1977/01/04	29.0	27.0	-99.9	-99.9	30.0	31.4	30.4
1977/01/05	29.6	28.8	-99.9	-99.9	28.8	31.0	30.0
1977/01/06	30.2	32.0	-99.9	-99.9	31.6	37.9	31.8

Cuadro 3.1: Ejemplo del formato de los datos recibidos. Las estaciones de medición son las numeradas del 11000 al 11008. Los campos son las temperaturas máximas. Y el $-99{,}9$ simboliza la indisponibilidad del dato.

En total, se tiene la siguiente información disponible:

- temperaturas máximas;
- temperaturas mínimas;
- precipitaciones; y
- estaciones de medición: identificador, nombre, país, ubicación geográfica y altitud.

En nuestro trabajo decidimos enfocarnos en los datos de las temperaturas máximas, por lo que las experimentaciones se realizaron en este conjunto.

3.1.3. Análisis descriptivo de los datos

Como ya mencionamos en la Sección 3.1.1, los datos de origen ya habían pasado por un proceso inicial de análisis, depuración y corrección. Realizamos nuestro propio proceso de análisis para tener un mejor conocimiento de la información y corregir algunos datos en caso de ser necesario. A continuación, se puede observar la información relevante del análisis realizado sobre el conjunto de temperaturas máximas, el cual es el conjunto de datos que usamos para la experimentación:

- **Rango de fechas:** 1977-01-01 al 2018-12-31. Total de 42 años (15340 días). No se detectó la pérdida de datos en alguna fecha en particular, es decir, es una secuencia completa de registros en el período indicado.
- **Total de estaciones:** 137.

- **Cantidad de registros faltantes (campos vacíos):** 100915 (representa el 4.8 % del total de 2101580 mediciones).

- **Valor más alto:** 330.20 °C, registrado en Viedma, Aeropuerto (Argentina) el 14 de enero de 1990. Podemos asumir que este registro es incorrecto debido al alto valor que tiene, y que probablemente haya ocurrido un error al medir o documentar. La segunda temperatura más alta registrada es de 54.4 °C, en el aeropuerto de Monte Caseros (Argentina), el 23 de febrero de 1983.

- **Valor más bajo:** -17.00 °C registrado en Maquinchao (Argentina) el 13 de julio de 1991.

- **Temperatura media:** 22.65 °C.

- **Desviación estándar:** 7.47 °C.

Respecto a la cantidad de datos, observamos que en comparación a los trabajos previos analizados, disponemos de una buena cantidad de datos, aunque no parece una cantidad suficiente para aplicar en forma directa en técnicas de aprendizaje profundo.

Al momento de hacer un análisis de los registros de temperatura, pero agrupando los datos por la estación de medición, observamos que:

- **Promedio de temperatura media de cada estación de medición:** 22.69 °C.

- **Promedio de desviación estándar de cada estación de medición:** 6.01 °C.

De esto se puede decir que si bien el registro de temperatura media no cambió, la desviación estándar al agrupar los datos por la estación de medición se redujo en más de un grado. Esto indica que la variación de temperaturas al analizar cada estación por separado es menor a la variación en todo el conjunto de datos. Por lo tanto, es de interés distinguir los puntos de medición según la información geográfica al momento de analizar las temperaturas.

El Cuadro 3.2 resume la información sobre las temperaturas agrupando por estaciones del año (verano, otoño, invierno y primavera) y la Figura 3.1 refleja la distribución de estos mismos datos. El gráfico en caja (*boxplot*) de más a la izquierda (bajo el nombre "Completo") toma en cuenta todo el conjunto de datos sin distinguir época del año ni ubicación geográfica, mientras que los siguientes manejan los datos de cada estación del año. Se puede observar que nuevamente la desviación estándar disminuye al especificar un punto geográfico y la época. Otra característica a tener en cuenta es que el rango de temperaturas, además de reducirse en tamaño, cambia los valores dependiendo de la fecha, lo cual tiene sentido, ya que en invierno las temperaturas son más bajas que en verano.

	Temperatura promedio	Desviación estándar
Verano	28.25	5.45
Otoño	22.63	6.37
Invierno	16.87	6.62
Primavera	22.96	6.69

Cuadro 3.2: Promedio temperatura máxima y desviación estándar agrupando por estaciones del año.

Figura 3.1: Gráfico en cajas de la distribución global de los registros, y los datos agrupados por estación del año.

Información faltante

Consideramos que un dato es faltante o nulo cuando el registro correspondiente a una fecha y estación específica no se encuentra en el conjunto de datos. La ausencia del mismo puede deberse, por ejemplo, a un error en el proceso de medición, en la recopilación o en el procesamiento, entre otros. Es importante contar con un conjunto de datos lo más completo posible, ya que permite trabajar mejor con los modelos de predicción. El Cuadro 3.3 recopila la cantidad de registros faltantes respecto al conjunto total de datos, indicando un 4,8 % (100.915 registros del total de 2.101.580), y la Figura 3.2 indica la proporción a escala al comparar los datos presentes frente a los faltantes.

Se puede apreciar que la cantidad de datos faltantes es relativamente baja respecto del total. Sin embargo, es importante identificar si esta ausencia se reparte por todo el conjunto o si se concentra en un período de tiempo o estación de medición en particular, ya que de ser así se deberían tomar medidas para no afectar el entrenamiento de los modelos.

Tipo de registro	Cantidad
Nulos	100915
No Nulos	2000665
Total	2101580

Cuadro 3.3: Resumen cantidad de registros nulos (faltante) y no nulos.

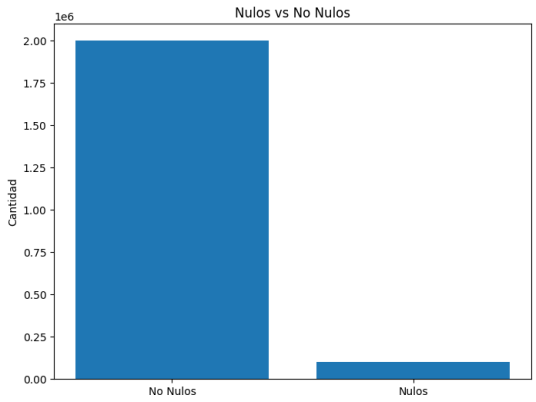

Figura 3.2: Comparación de datos faltantes respecto de los presentes en el conjunto total de datos, respecto a mediciones de temperaturas máximas.

La Figura 3.3 hace un análisis de la falta de información agrupando los datos en periodos de 5 años. Allí se puede observar como la ausencia de información es uniforme en todo el período de tiempo, y que esta es pequeña respecto al conjunto total en el mismo período.

Al revisar los datos agrupando por estación de medición se detectaron los siguientes datos respecto a la información faltante: la estación con mayor cantidad de datos nulos carece de un 20 % de los datos. Luego le siguen 21 estaciones con información faltante entre el 10 % y el 20 %. Por lo tanto, solamente 22 estaciones del total de 137 tienen más del 10 % de registros vacíos.

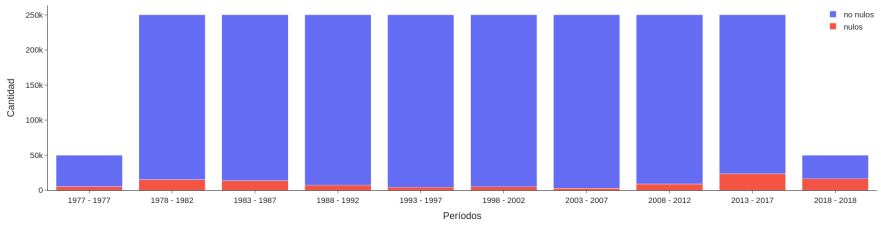

Figura 3.3: Cantidad de información nula respecto al total del conjunto de datos de temperaturas máximas, agrupando en periodos de 5 años.

Teniendo en cuenta este análisis, podemos concluir que la falta de información en el conjunto de datos originales no es significativa y que no debería tener gran impacto en la performance de los modelos entrenados y evaluados, ya que esta ausencia se encuentra distribuida entre todo el tiempo y espacio manejado. Sin embargo, es importante tener presente la distribución de esta anomalía y

los porcentajes al momento de partir el conjunto de datos en conjuntos más pequeños de experimentación.

Análisis geográfico

Tenemos información de temperaturas provenientes de estaciones de los siguientes países: Argentina, Brasil, Chile, Paraguay y Uruguay. En la Figura 3.4 se puede ver la distribución de las estaciones en el mapa.

Figura 3.4: Mapa con distribución de las estaciones de medición usadas para relevar la información.

Con el objetivo de determinar si existe cierta relación entre las temperaturas y la ubicación geográfica, se determinaron las temperaturas extremas en las estaciones y se buscaron relaciones. Definimos las temperaturas extremas de una estación a aquellas que se encuentren dentro del percentil noventa del registro de temperaturas correspondiente a la misma. En el Cuadro 3.4 se puede observar el resultado de dicho análisis, destacando las 5 estaciones con el mayor promedio de temperaturas máximas extremas. A su vez, la Figura 3.5 refleja la ubicación de dichas estaciones. Se puede observar que las mismas se encuentran en el extremo norte del conjunto.

Ranking	Estación	País	Long	Lat	Altura	Promedio
1	Mariscal-Estigarribia-Aero	Paraguay	-60.62	-22.03	167.0	40.18
2	Las Lomitas	Argentina	-60.35	-24.42	130.0	39.49
3	La Rioja Aero	Argentina	-66.50	-29.23	429.0	39.17
4	Santiago del Estero Aero	Argentina	-64.18	-27.46	199.0	38.89
5	Puerto-Casado	Paraguay	-57.94	-22.28	87.0	38.54

Cuadro 3.4: Resumen de las 5 primeras estaciones con el mayor promedio de temperaturas máximas extremas.

Figura 3.5: Mapa ubicando las estaciones con mayor promedio de temperaturas extremas.

Finalmente, la Figura 3.6 agrupa las estaciones en tres conjuntos según el promedio de las temperaturas extremas: donde los puntos verdes representan los resultados más bajos, seguidos por los puntos amarillos, y finalmente los puntos rojos representan el conjunto de estaciones con mayor promedio de temperaturas máximas. Se puede observar como estas últimas se encuentran distribuidas en el centro del continente, mientras que el subconjunto de datos de valores más bajos están lo más próximo a las costas oceánicas. Por lo tanto, conocer la estación de medición, su ubicación geográfica, y las temperaturas de las estaciones cercanas es información útil y fuertemente relacionada con la ocurrencia de sus temperaturas.

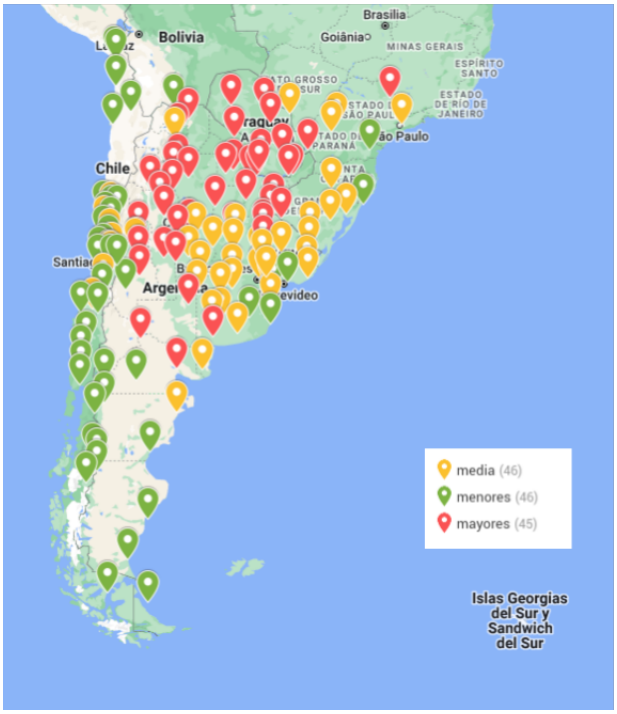

Figura 3.6: Estaciones de medición meteorológica.

3.1.4. Depuración y pre-procesamiento

Previo a la etapa de experimentación, llevamos a cabo una fase de preprocesamiento de datos en forma iterativa, donde realizamos un análisis de los datos, aplicamos transformaciones e imputamos datos faltantes.

Preprocesamiento de precipitaciones

En el conjunto de datos inicial, los datos de temperaturas máximas y los de precipitaciones se encontraban separados, por lo que los unificamos en un único conjunto. A su vez, el conjunto de precipitaciones tenía datos faltantes, los cuales imputamos reemplazándolos por las precipitaciones del día anterior.

Preprocesamiento de temperaturas máximas

Como ya mencionamos en la Sección 3.1.3, el conjunto de datos de temperaturas máximas contaba con varios datos faltantes. Para el manejo de los mismos, tuvimos en cuenta las particularidades de los modelos que utilizamos para la experimentación. Una de estas particularidades es que *Graph WaveNet* requiere que sus datos estén completos, es decir, no maneja datos vacíos o nulos. Esto es distinto para el caso de *XGBoost*, el cual soporta este tipo de datos e incluso

se indica que logra obtener buenos resultados en su presencia. Inicialmente, se plantearon dos versiones de conjuntos de datos: una en la que reemplazáramos los datos faltantes y otra en la que los mantuviéramos, y así experimentar con *Graph WaveNet* y *XGBoost* respectivamente (aprovechando las ventajas de este último). Sin embargo, la segunda versión resultaba con un desbalance que perjudicaría la capacidad de medir la performance de los modelos, por lo que optamos por utilizar solo una versión del conjunto de datos donde los datos faltantes se imputaron reemplazándolos por las mediciones del día anterior de la misma estación meteorológica.

Formato de los datos de entrada

Otra particularidad de los modelos que utilizamos en la experimentación es el formato de los datos que reciben. *XGBoost* recibe un formato tabular basado en características, en el que en cada fila se identifica la observación de una fecha y estación en particular y se agregan un conjunto de atributos predictores (*features*). El Cuadro 3.5 refleja un ejemplo de entrada para *XGBoost*.

id estación	**fecha**	feature 1	feature 2	feature 3	feature 4
1	**1999-10-01**	30	29	30	28
2	**1999-10-01**	25	27	26	25

Cuadro 3.5: Ejemplo de la disposición tabular de los datos para *XGBoost*.

Por otro lado, el modelo desarrollado por los autores de *Graph WaveNet* espera datos con un mayor grado de procesamiento. El modelo recibe una lista con elementos de dimensiones $t \times N \times D$, donde t es la cantidad de mediciones del pasado o futuro que se quiere usar para la predicción, N es la cantidad de estaciones de medición (137 en nuestro caso) y D es la cantidad de *features* que se utilizan para la predicción. Es decir, cada elemento de la lista de entrada contendrá, para cada t día del pasado, las D características de las 137 estaciones.

En la Figura 3.7 podemos ver un ejemplo del formato de un elemento de entrada de *Graph WaveNet*. Allí asumimos que se quiere usar una ventana de 12 días de información del pasado.

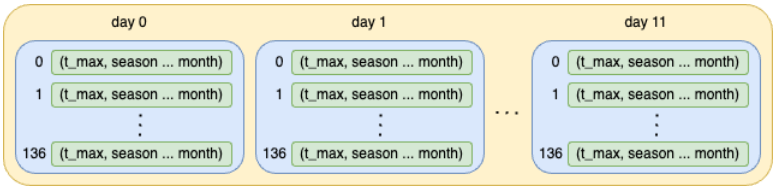

Figura 3.7: Ejemplo de un elemento de entrada de *Graph WaveNet* para una ventana de tiempo de 12 días.

Para obtener el formato descrito, desarrollamos un *script* que transforma el conjunto de datos tabulares iniciales. Para el mismo usamos como base el código

que implementaron los autores, el cual adaptamos a nuestro problema.

3.1.5. Separación de los datos en *train*, *validation*, y *test*

Previo a manipular el conjunto de datos de forma particular en cada técnica, dividimos el conjunto en 3 partes:

- **train:** usado para que el modelo entrene, es decir, que optimice sus parámetros para mejorar las predicciones;
- **validation:** usado durante el entrenamiento para evaluar la calidad del proceso, y en especial, para que no haya sobre-ajuste a los datos de entrenamiento; y
- **test:** usado para medir la calidad del modelo con un conjunto de datos nuevo.

En general se suele asignar datos de forma des-balanceada entre estos tres conjuntos, aportándole más ejemplos al conjunto de *train*, luego al de *test* y finalmente al de *validation*. Como es habitual en los problemas con dependencias temporales, agrupamos los datos por conjuntos de años, de forma tal que se tiene una proporción equitativa de las cuatro estaciones del año en todos los conjuntos. La división fue de un 60 % para *train*, un 20 % para *test* y un 20 % para *validation*. Disponiendo de un total de 42 años, y redondeando para agrupar la misma cantidad de días, se tiene: 26 años para *train*, luego 8 años para *validation* y finalmente 8 años para *test*. Como resultado se tiene que los datos se reparten de la siguiente forma:

- **train**: del 01/01/1997 al 31/12/2002;
- **validation**: del 01/01/2003 al 31/12/2010; y
- **test**: del 01/01/2011 al 31/12/2018.

3.1.6. Matriz distancia

Además del trabajo realizado sobre el conjunto de datos de las temperaturas, hicimos el cálculo de la distancia geodésica entre todas las estaciones de medición, con el objetivo de dimensionar la proximidad entre ellas[10]. La distancia geodésica es la distancia sobre la curva de la superficie terrestre, teniendo en cuenta la latitud y longitud entre dos puntos. Como resultado se tiene una matriz de dimensiones 137x137 con las filas y columnas indexadas por las estaciones, y cada celda registra la distancia entre la estación identificada por la fila y columna correspondiente.

En el siguiente capítulo se observará que esta matriz será usada de información para mejorar los modelos.

[10]Para calcular la distancia geodésica usamos la librería de *Python* llamada *Geopy* https://geopy.readthedocs.io/en/stable/

3.2. Metodología para construir los modelos predictivos

En esta sección desarrollamos el trabajo llevado adelante para la construcción de los modelos utilizados: *Graph WaveNet* (Wu y cols., 2019) y *XGBoost* (Chen y Guestrin, 2016). Ambos manejaron la misma información (cada uno con algunos cambios en el formato, dependiendo de la necesidad del modelo). Inicialmente, se planteó el mismo objetivo: dado información meteorológica de doce días en el pasado e información geográfica, se desea saber la temperatura máxima de cada estación para el día siguiente. Este fue definido por facilidad de la arquitectura y evitar dificultades en las primeras etapas. Teniendo en cuenta esta finalidad, se construyeron distintos modelos que fueran perfeccionándose hasta llegar a un punto que consideramos aceptable. En esta sección detallamos la metodología para construir estos modelos. En la Sección 4 se presentan resultados sobre otros experimentos, donde se utilizan otros periodos de datos del pasado y también se predicen más de un día en el futuro.

3.2.1. Medidas de evaluación a usar

Como se vio en la Sección 2.3, existen distintas medidas usadas para calcular la exactitud de los modelos cuando se tratan problemas de series de tiempo. Nosotros optamos por medir el rendimiento de los modelos frente a los conjuntos de evaluación y testeo con la Raíz del Error Cuadrático Medio (*RMSE* por sus siglas en inglés). Esto es porque nos parece sencilla la interpretación de como mejoran los modelos y su interpretación del error en la misma unidad que el valor a predecir.

3.2.2. Ingeniería de características (*features*)

La ingeniería de características es el proceso de obtención de nuevos atributos a partir de los ya conocidos, con el objetivo de mejorar la predicción. Esto se logra mediante la aplicación de transformaciones matemáticas a otros atributos. Es un proceso fundamental, ya que el desempeño de la mayoría de los modelos de aprendizaje automático depende de la representación de su vector de atributos de entrada (Heaton, 2016).

El método que adoptamos para este proceso es de generar una característica nueva, introducirla en el conjunto de datos de entrenamiento, y decidir si incorporarla definitivamente de acuerdo al desempeño del modelo respecto al *RMSE* en el conjunto de validación. En nuestro caso, y según el conjunto de datos con el que contamos, podemos generar características nuevas que introduzcan información de tres tipos: temporal, espacial y meteorológica.

- Para las características temporales podemos usar la información de la fecha de cada medición para generar datos un poco más explícitos sobre el tiempo. Algunos ejemplos de características temporales que pueden resultar interesantes son mes, estación del año o día del año.

- Como posibles características espaciales podemos utilizar la información de latitud y longitud para poder generar datos espaciales nuevos.
- Como posibles características meteorológicas podemos usar la información sobre las precipitaciones.

Este proceso varía según el modelo de aprendizaje automático con el que estemos trabajando. Es importante destacar que, a diferencia de *Graph WaveNet*, *XGBoost* requiere fuertemente de un proceso previo de ingeniería de características para tener un conjunto de datos como entrada del modelo, ya que este maneja la información en forma tabular, y no es capaz de inferir datos nuevos a partir del conjunto dado. Este proceso es menos requerido en *Graph WaveNet*, ya que es capaz de aprender las interrelaciones temporales y espaciales.

A la hora de elegir qué características nuevas generar, debemos tener en cuenta el formato de cada modelo y cómo es el tipo de características que acepta. Por un lado, *Graph WaveNet* acepta únicamente series de tiempo numéricas, por otro lado, *XGBoost* acepta cualquier tipo de características numéricas. Por ejemplo, si usamos las precipitaciones como característica meteorológica; en el caso de *Graph WaveNet* podría usarse como serie de tiempo, y en el caso de *XGBoost* podría usarse como índices exógenos y general a cada estación de medición.

3.2.3. Configuración de hiper-parámetros

Los hiper-parámetros son parámetros de un modelo configurados previo a ejecutar el proceso de entrenamiento. Estas configuraciones indican el comportamiento que toma el modelo mientras entrena. Todos los modelos de aprendizaje automático tienen hiper-parámetros, los cuales son inicializados con un valor por defecto.

El proceso de configuración de hiper-parámetros (Hutter, Kotthoff, y Vanschoren, 2019) se limita a variar estos parámetros de forma de optimizar el modelo para el problema que se busca resolver. Hoy en día existen procesos automáticos que facilitan esta tarea. Es importante realizar esta tarea de configuración porque ahorra tiempo a las personas para trabajar con modelos de aprendizaje automático, y además perfila el modelo creado a la realidad en la que se está trabajando.

3.2.4. Baseline

Al igual que en algunos de los trabajos mencionados en la Sección 2.3, se definió un modelo básico para determinar una línea base de rendimiento para luego comparar con los modelos (*Graph WaveNet* y *XGBoost*). Para ello creamos un modelo llamado *Baseline*, donde los modelos de mayor complejidad se consideran 'aceptables' si logran un mejor rendimiento que este.

Este modelo *Baseline* funciona de la siguiente forma: la predicción del día siguiente será la temperatura del día actual. Es decir, retorna 'sin cambios' la temperatura. Este razonamiento se fundamenta en que por lo general las

temperaturas no sufren grandes cambios de un día al siguiente. La Figura 3.8 refleja el funcionamiento del modelo planteado.

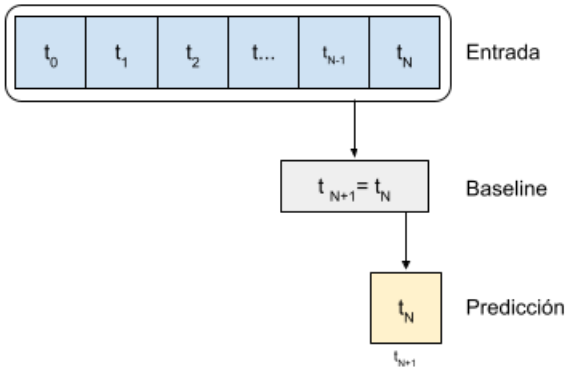

Figura 3.8: Ejemplo de modelo *Baseline* planteado: la predicción de la temperatura del día siguiente es la misma que la del día actual.

La Figura 3.9 ejemplifica el funcionamiento del modelo usando el conjunto de datos. Allí se puede ver las mediciones de temperatura en la estación 11000 (Artigas, Uruguay) del día 1/1/2011 al 12/1/2011 como información de entrada, y también la lógica usada para la predicción del 13/1/2011 (copiando el registro del último día). Además, se puede ver el valor real correspondiente para la fecha y estación de medición manejada, en este caso el *RMSE* es 0.89 °C.

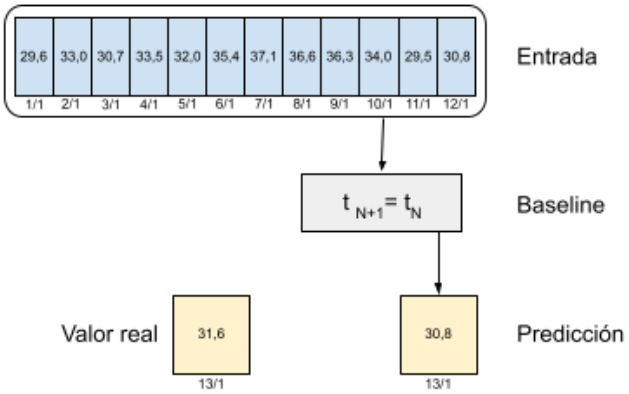

Figura 3.9: Ejemplo de modelo *baseline* planteado basándose en el período de 1/1/2011 al 12/1/2011 en la estación 11000.

Este modelo es ejecutado y evaluado bajo el mismo conjunto de datos que

los otros modelos, y en particular usamos la misma estructura de datos usada por *XGBoost*, ya que, como se detallará en la siguiente sección, resulta sencillo obtener la salida esperada a partir de esta.

3.2.5. XGBoost

Medida de evaluación en el entrenamiento

Optamos por medir el error con *RMSE* durante el proceso de entrenamiento de los modelos *XGBoost*, ya que es la misma que se va a usar al evaluar los modelos en la etapa de testeo.

Hiper-parámetros

Como desarrollamos en la Sección 2.5, *XGBoost* cuenta con una gran cantidad de hiper-parámetros configurables para la creación y entrenamiento de modelos[11]. A continuación listamos en el Cuadro 3.6 aquellos que consideramos de mayor relevancia, que incluyen: el funcionamiento general, las optimizaciones en los árboles de decisión, los parámetros de aprendizaje, entre otros.

Nombre	**Descripción**	**Defecto**	**Rango**
n_estimators (num_round)	Número de árboles a generar	100	[0,∞]
booster	Técnica de *boosting* a usar	gbtree	[gbtree, gblinear, dart]
learning_rate (eta)	Coeficiente para graduar el tamaño de cada árbol al objetivo	0.3	[0,1]
min_split_loss (gamma)	Usado para decidir la división y podado de arboles	0	[0,∞]
max_depth	Largo máximo de un árbol	6	[0,∞]
min_child_weight	Peso mínimo asignable a un hijo en el árbol	1	[0,∞]
subsample	Porcentaje de división del conjunto de datos	1	(0,1]
sampling_method	Método de división de los datos	uniform	[uniform, gradient_based]
*colsample_by**	Familia de parametros (bytree, bylevel y bynode) que indican cuantas caracteristicas se usan	1	(0,1]
lambda (reg_lambda)	Parámetro de regularización	1	[0,1]
alpha (reg_alpha)	Parámetro de regularización	0	[0,1]
tree_method	Técnica de construcción del árbol	auto	[auto, exact, approx, hist, gpu_hist]
grow_policy	Política de decisión para crear nodos	depthwise	[depthwise, lossguide]
max_leaves	Cantidad máxima de hojas en un árbol	0	[0,∞]
objective	Tipo de función de regresión a usar	reg:squarederror	variado, se destacan reg:squarederror, reg:squaredlogerror,reg:gamma
eval_metric	Métrica para evaluar la función de perdida	rmse (en regresión)	variado, se destacan: rmse, mae, logloss, error
seed	Valor randómico agregado al algoritmo	0	[0,∞]

Cuadro 3.6: Parte del conjunto de parámetros configurables para *XGBoost*.

Al momento de ejecutar un proceso de configuración de hiper-parámetros para *XGBoost*, se seleccionan los parámetros a optimizar para el problema particular, y se define un rango de valores para cada uno de estos parámetros. Entrenamos un nuevo modelo con cada combinación de parámetros seleccionada y lo evaluamos sobre el conjunto de evaluación. Finalmente, nos quedamos con aquel modelo que resulte con menor medida *RMSE*.

[11] https://xgboost.readthedocs.io/en/stable/parameter.html

3.2.6. Graph WaveNet

Medida de evaluación en el entrenamiento

Para el caso de *Graph WaveNet*, se utiliza la métrica *MAE* como función de pérdida (*loss*) a la hora de ajustar en su entrenamiento. Esto es algo que ya viene implícitamente definido como parte de la arquitectura del modelo.

Hiper-parámetros

Graph WaveNet al igual que la mayoría de modelos de aprendizaje automático, posee un conjunto de hiper-parámetros que determinan una posible conformación del modelo, y determinan de qué forma es que el mismo entrena, aprende y ajusta. Los hiper-parámetros de *Graph WaveNet* son:

- *hidden nodes*;
- *input channels*;
- *number of nodes*;
- *batch size*;
- *learning rate*;
- *dropout*;
- *weight decay*; y
- *epochs*.

Entendemos que *learning_rate*, *epochs*, *batch_size*, *weight_decay* y *dropout* son los hiper-parámetros que pueden generar un mayor impacto en el desempeño del modelo, por lo que los consideramos esenciales a la hora de llevar a cabo un proceso de configuración de hiper-parámetros. Para esta etapa, definimos rangos de valores posibles para cada parámetro y evaluaremos el *RMSE* y el *MAE* del modelo para cada una de las combinaciones de hiper-parámetros.

Matrices de adyacencia

Como ya vimos en la Sección 2.4.2, la arquitectura *Graph WaveNet* hace uso de una matriz de adyacencia para el manejo de la información espacial del problema. La misma puede tener distintas estructuras dependiendo de cómo se la configure y de la información espacial que se encuentre disponible.

Los autores identifican los siguientes tipos de matrices:

- ***Identity***: Matriz de identidad, podemos escribirla como $[\boldsymbol{I}]$.
- ***Forward-only***: definida en la Ecuación (2.11), podemos escribirla como $[\boldsymbol{P}]$.

- **Adaptive-only**: definida en la Ecuación (2.14), podemos escribirla como $[\tilde{\mathbf{A}}_{adp}]$.
- ***Forward-backward***: definida en la Ecuación (2.12), podemos escribirla como $[\mathbf{P}_f, \mathbf{P}_b]$.
- ***Forward-backward-adaptive***: definida en la Ecuación (2.15), podemos escribirla como $[\mathbf{P}_f, \mathbf{P}_b, \tilde{\mathbf{A}}_{adp}]$.

Vale aclarar que todas estas opciones de matrices son simétricas y en nuestro caso de dimensión 137x137, donde cada columna y fila hace referencia a una estación de medición.

Para hacer uso de estas distintas configuraciones de matrices, tenemos a disposición los parámetros *adjdata*, *adjtype*, *aptonly*, *addaptadj* y *randomadj*, los cuales variamos para lograr el tipo de matriz deseada.

Capítulo 4

Experimentación

A continuación desarrollamos el trabajo realizado y los resultados obtenidos en la construcción y evaluación de modelos. Como comentamos en la Sección 3.2.1, la principal métrica a tomar en cuenta para medir la correctitud de los modelos construidos es la raíz del error cuadrático medio (*RMSE*, por sus siglas en inglés *Root Mean Squared Error*), pero también tomamos en cuenta otros factores como ser: los tiempos de entrenamiento y testeo de los modelos, y el comportamiento de las métricas de pérdida durante el proceso de entrenamiento.

Usamos el lenguaje Python[1] para la implementación del código, tanto para el procesamiento de datos como para la creación y entrenamiento de modelos. Para la etapa de experimentación usamos distintos entornos de trabajo, los cuales fuimos manejando a medida que requeríamos otras configuraciones o mayor capacidad de cómputo. Los entornos utilizados fueron:

- Entornos locales usando entornos virtuales de Python para instalación de paquetes.

- Google Colab[2], permitiendo compartir de forma remota el trabajo y accediendo a mejores recursos de cómputo. En particular se tiene acceso a procesadores gráficos (GPU), lo que es requisito para entrenar los modelos *XGBoost* y *Graph WaveNet*.

- ClusterUy[3] (Nesmachnow y Iturriaga, 2019) usado en particular para la configuración de hiper-parámetros debido a que este permite la ejecución de código en tiempos mayores al entorno de Google Colab y pone a nuestra disposición recursos como GPU.

[1] https://www.python.org, accedido el 06-10-2022
[2] https://colab.research.google.com, accedido el 06-10-2022
[3] https://www.cluster.uy, accedido el 06-10-2022

4.1. Baseline

La implementación de este modelo *Baseline* no implica grandes complejidades ni la ejecución de un entrenamiento o configuración de hiper-parámetros, ya que simplemente se retorna uno de los valores de entrada. Por lo tanto, prescindimos del conjunto de datos de validación.

Optamos por manejar los datos en el mismo formato que generamos para *XGBoost*, ya que este va a resultar de forma directa la temperatura de cada estación del día anterior como temperatura predicha para el próximo día.

Al evaluar este modelo sobre el conjunto de datos de testeo, y medir el error con la métrica *RMSE*, obtenemos una medida de 3,508 °C.

4.2. XGBoost

4.2.1. Resultados intermedios

Tal como comentamos sobre el final de la Sección 2.5, *XGBoost* es un modelo que requiere se le indique de forma explícita las características a utilizar, es decir, no tiene la capacidad para inferir nuevas características durante el aprendizaje. Por lo tanto, este trabajo es de vital importancia para tener un buen rendimiento en los modelos de esta arquitectura.

Definimos un modelo inicial basado en características meteorológicas de los datos. Las primeras características agregadas corresponden a la ventana de tiempo de entrada (inicialmente de 12 días), con los registros de temperaturas máximas de cada estación. Es decir, para cada fecha y estación dada se quiere obtener el histórico de las temperaturas correspondientes de esa estación en el pasado. Llamamos x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} y x_{11} a las distintas características extraídas, donde x_0 indica la temperatura 12 días antes del día a predecir y x_{11} el registro del día anterior. Todas de tipo numérico. Luego de entrenar un modelo con este conjunto de datos de entrada y usar el conjunto de datos de validación, obtenemos un *RMSE* inicial de 3.219139 °C.

Otro parámetro de interés es la información de registros de precipitaciones diaria en cada estación, ya que manejamos la hipótesis de que las lluvias de un día influyen en la temperatura del ambiente. Agregamos las mediciones de cantidad de precipitaciones (definida como *pp*, de tipo numérico) correspondiente al último día de la ventana de tiempo de entrada.

A partir de todo el histórico de precipitaciones es posible obtener una variable exógena para cada estación en el rango de fechas disponible denominada Índice Estandarizado de Precipitación[4] (SPI, por sus siglas en inglés: Standardized Precipitation Index). El SPI es un indicador que representa el total de mediciones de precipitaciones para un lugar en períodos largos de tiempo. Agregamos los datos $spi30$, $spi90$, $spi180$, $spi270$ y $spi360$ de tipo numéricos, los cuales representan los índices SPI de precipitación acumulada del conjunto de

[4] https://climate-adapt.eea.europa.eu/en/metadata/map-graphs/spi-standardized-precipitation-index, accedido el 02-04-2023

30, 90, 180, 270 y 360 días (hasta el último día de la ventana de tiempo de entrada). El nuevo modelo entrenado con estos parámetros disminuye el *RMSE* sobre el conjunto de validación a 3.105949 °C.

Otro tipo de información que resulta interesante como entrada a un modelo de serie de tiempo es aquella que aporte datos temporales al conjunto, como puede ser la que identifique en que momento del año ocurre la misma. *XGBoost* no acepta datos de tipo fecha o tipo texto como entrada, debido a que no es capaz de determinar una regla de orden entre cada uno de estos datos, por tanto, no puede agregarse directamente la fecha como atributo. Debido a esto, buscamos otras alternativas que logren representar la misma información. Agregamos los parámetros *season* y *day_of_year* con motivo de brindarle al modelo cierta información que le permitan comprender cuáles son los rangos de temperatura según el momento del año. Estos parámetros indican la estación del año en la que ocurrió la medición (verano, otoño, invierno o primavera) y el número de día en el año. Para el parámetro *season* agrupamos de a tres meses asignando un número entre cero y tres a las estaciones del año:

- 0 representando el verano con los meses diciembre, enero y febrero;
- 1 representando el otoño con los meses marzo, abril y mayo;
- 2 representando el invierno con los meses junio, julio y agosto; y
- 3 representando la primavera con los meses setiembre, octubre y noviembre.

Luego el parámetro *day_of_year* es inicializado con el número en el año correspondiente a la fecha, teniendo en cuenta que el primero de enero se le asigna el 1, y de forma consecutiva hasta llegar al 31 de diciembre con el 365. También tenemos en cuenta los años bisiestos, en donde la secuencia va de 1 a 366.

Un detalle a tener en cuenta es que el parámetro *season* lo inicializamos como variable categórica[5] (Distributed (Deep) Machine Learning Community, 2023) en lugar de numérica. Las variables categóricas se diferencian de las numéricas en que manejan un dominio discreto y finito en lugar de uno continuo e infinito, como lo hacen las segundas. De esta forma evitamos confundir al modelo con un orden inexistente entre las estaciones del año, y tampoco se brinda una noción errónea de cercanía entre ellas. Una posible implementación para representar a una variable categórica es usando la notación *one hot encoding* (Seger, 2018), sin embargo, en la versión 1.6.0 de *XGBoost* para Python liberada en mayo de 2022 [6] soporta este tipo de variables. A diferencia de *season*, la variable *day_of_year* no la inicializamos como categórica, ya que es de interés indicar la noción de cercanía entre fechas. Por ejemplo: las temperaturas en verano (con un índice de 1, 2, 3,... según *day_of_year*) se encuentran relacionadas, y tienen un rango de valores distinto a aquellas en invierno (con un índice 182,183,...).

[5] *Understand your dataset with XGBoost*: https://cran.r-project.org/web/packages/xgboost/vignettes/discoverYourData.html, accedido el 20-03-2023

[6] https://github.com/dmlc/xgboost/releases/tag/v1.6.0

También intentamos agregar más información como el número de día, mes y año, cada dato por separado. Sin embargo, no lograron mostrar resultados favorables, por lo que optamos por eliminarlas.

El nuevo modelo entrenado con los parámetros temporales *season* y *day_of_year* disminuye el *RMSE* sobre el conjunto de validación a 3.080650 °C.

Sumado a la identificación de cada estación de medición, queremos agregar al modelo cierta noción de ubicación geográfica de las estaciones, además de la capacidad de identificar a qué estación pertenece cada registro de medición. En otras palabras, además de reconocer el lugar físico donde ocurre la medición, se pretende aportar información de lo que ocurre en el contexto geográfico. Como el objetivo es predecir temperaturas, agregamos las mediciones de las tres estaciones más cercanas a la que se está manejando. Estas estaciones las llamamos 'vecinos'. Logramos identificar dichas estaciones y sumar la temperatura máxima de las mismas correspondiente al último día en la ventana de tiempo. Esto lo logramos utilizando la matriz de distancias generada basándonos en la latitud y longitud de cada estación comentada en la Sección 3.1.6. Nombramos entonces las variables n_1, n_2 y n_3 a las variables que registran las temperaturas máximas para los vecinos de cada estación, en la ventana de tiempo manejada. Estas variables son de tipo numérico.

Otro parámetro que intentamos agregar en cada registro fue la información de latitud y longitud de cada estación. Estos datos no aportaron mejoras al modelo entrenado. Se puede especular que esto se debe a que no aportan más información que la que ya hacía el identificador de estación.

Luego de correr el proceso de entrenamiento, al agregar estas últimas características, se obtiene un modelo que al evaluarlo en el conjunto de validación se mide un *RMSE* de error de 2.947073 °C, lo cual indica una mejora frente al modelo anterior. Este modelo resulta el que muestra mejores resultados durante la experimentación y evaluamos en el conjunto de testeo. El Cuadro 4.1 lista las características finales obtenidas durante este proceso de ingeniería de características junto a su tipo de dato. El Cuadro 4.2 resume los resultados de los modelos desarrollados en esta sección, el *RMSE* al evaluar frente al conjunto de validación, y los tiempos de entrenamientos asociados. Allí se puede notar como la medida de evaluación disminuye cuando se agregan nuevas características, y a su vez el tiempo de entrenamiento crece.

4.2.2. Configuración de hiper-parámetros

En las primeras etapas experimentales no indicamos particularidades en los hiper-parámetros de entrenamiento para *XGBoost*, sino que mantuvimos todos sus valores por defecto, ya que se indicaban como los recomendados y presentaron resultados aceptables desde los primeros experimentos (es decir, igualaban o mejoraban el rendimiento del modelo *Baseline*). Sin embargo, tuvimos que configurar ciertas particularidades para que el modelo sea capaz de manejar variables categóricas:

- *tree_method* = 'gpu_hist';

Parámetro	Descripcion	Numérico
$station\_id$	Identificador de estación	Categórico
x_0	Registro de temperatura hace 12 días	Numérico
x_1	Registro de temperatura hace 11 días	Numérico
x_2	Registro de temperatura hace 10 días	Numérico
x_3	Registro de temperatura hace 9 días	Numérico
x_4	Registro de temperatura hace 8 días	Numérico
x_5	Registro de temperatura hace 7 días	Numérico
x_6	Registro de temperatura hace 6 días	Numérico
x_7	Registro de temperatura hace 5 días	Numérico
x_8	Registro de temperatura hace 4 días	Numérico
x_9	Registro de temperatura hace 3 días	Numérico
x_{10}	Registro de temperatura hace 2 días	Numérico
x_{11}	Registro de temperatura en el día anterior	Numérico
pp	Registro de precipitaciones en el día anterior	Numérico
$spi30$	SPI del día anterior respecto de los últimos 30 días	Numérico
$spi90$	SPI del día anterior respecto de los últimos 90 días	Numérico
$spi180$	SPI del día anterior respecto de los últimos 180 días	Numérico
$spi270$	SPI del día anterior respecto de los últimos 270 días	Numérico
$spi360$	SPI del día anterior respecto de los últimos 360 días	Numérico
$season$	Identificación de la estación del año	Categórico
day_of_year	Día del año	Numérico
n_1	Temperatura de la estación más cercana en el día anterior	Numérico
n_2	Temperatura de la segunda estación más cercana en el día anterior	Numérico
n_3	Temperatura de la tercera estación más cercana en el día anterior	Numérico
y	Valor a predecir	Numérico

Cuadro 4.1: Estructura de datos de entrada del modelo final usada por *XGBoost*.

Conjunto de datos	T	S	RMSE	T. de entr.
Inicial: info. meteorólogica	12	1	3.106 °C	1.832 s
Info. meteorológica y temporal	12	1	3.081 °C	2.471 s
Info. meteorológica, temporal y espacial	12	1	2.947 °C	4.172 s

Cuadro 4.2: Resultados de modelos intermedios durante el proceso de ingeniería de características, evaluando el conjunto de validación.

- *enable_categorical* = True; y
- *use_label_encoder* = False

Es importante indicar que el uso del método *gpu_hist* para la construcción de árboles de decisión requiere exclusivamente del uso de procesadores gráficos. El uso de estos fue notorio al comparar el entrenamiento usando CPUs, en lo que respecta a tiempos de entrenamiento y ejecución.

Para el caso de la configuración de hiper-parámetros de *XGBoost* usamos la herramienta *Optuna*[7]. *Optuna* es una librería que permite la automatización de la configuración de hiper-parámetros para modelos de aprendizaje automático. Aplica tanto para funciones de maximización como minimización. Para lograr esto, usa distintos algoritmos de optimización de hiper-parámetros que se encuentran en el estado del arte y tiene la capacidad de descartar combinaciones candidatas que no indican ser prometedoras. Es fácil agregar la librería a un proyecto Python, donde se indican los parámetros a buscar para minimizar una función especificada y la cantidad de iteraciones a experimentar.

Luego de realizar el proceso de *feature engineering* e identificar las características de entrada para el modelo con un error aceptable usando el conjunto de datos de validación, procedemos a realizar la configuración de hiper-parámetros del modelo de *XGBoost* con el mismo conjunto de datos y con las características de entrada definidas. El Cuadro 4.3 resume los valores de hiper-parámetros seleccionados para el proceso de búsqueda de la mejor configuración que se ajuste al problema. En ella se puede ver el rango de búsqueda para cada parámetro, si el dominio de cada rango se maneja en un conjunto discreto o continuo (en caso de ocurrir el segundo se indica el tamaño del salto, es decir, la distancia entre los valores de evaluación) y finalmente el mejor valor obtenido. Para todos ellos, indicamos rangos de valores a explorar en torno a los valores por defecto, ya que en primeras iteraciones observamos que estos logran mejores resultados en comparación a rangos alejados a estos.

Para llevar adelante esta tarea usamos el entorno del *ClusterUy* donde se puede ejecutar este proceso de búsqueda con más de 14.400 iteraciones (es decir, posibles combinaciones dentro de todas las posibles permutaciones de los parámetros a experimentar en la configuración). El proceso total de optimización de hiper-parámetros en esta infraestructura necesitó de 4 días y medio para ser completado. Como resultado, obtenemos una configuración cuya evaluación al conjunto de validación refleja un *RMSE* de 2.908 °C, menor al registrado en etapas anteriores. La configuración de hiper-parámetros de *XGBoost* que resulta óptima en este problema se puede observar en el Cuadro 4.3. Los valores allí se encuentran redondeados para claridad.

4.2.3. Resultados con el modelo final

A continuación se muestran las métricas obtenidas para el modelo *XGBoost* que mostró mejores resultados, luego de realizarse la configuración de hiper-

[7] https://optuna.org, accedido el 06-10-2022

Parámetro	Rango	Dominio	Valor Final
n_estimators	(75, 125)	discreto	125
subsample	(0.2, 1.0)	continuo, pasos de 0.10	1.0
max_depth	(2, 9)	discreto	9
eta	(1e-08, 0.990)	continuo, pasos de 0.01	0.070
gamma	(1e-08, 0.990)	continuo, pasos de 0.01	0.260
alpha	(1e-08, 0.990)	continuo, pasos de 0.01	0.540
lambda	(1e-08, 1.490)	continuo, pasos de 0.01	0.930
colsample_bytree	(0.2, 1.0)	continuo, pasos de 0.1	1.0
min_child_weight	(1e-08, 1.990)	continuo, pasos de 0.01	0.160
grow_policy	[depthwise, lossguide]	discreto	lossguide

Cuadro 4.3: Hiperparámetros usados para *XGBoost*.

parámetros.

Importancia de los parámetros

XGBoost ofrece una funcionalidad en los modelos entrenados de devolver la ponderación que le asigna a cada una de las características usadas como entrada al momento de calcular la predicción para un conjunto de valores dado. Esta herramienta es de utilidad para comprender a grandes rasgos cómo es que maneja la información para tomar una decisión, y para validar su funcionamiento. La Figura 4.1 representa el gráfico de la importancia asignada a cada uno de los valores manejados para el modelo entrenado luego de la configuración de hiper-parámetros. Aunque no se pueda apreciar en la imagen por razones de escala, todos los parámetros manejan un valor mayor a cero y menor a uno. En ella se puede destacar como el modelo se basa principalmente (pero no de forma exclusiva) en la temperatura del día anterior para la predicción. Con esto se logra validar la correctitud en la elección del día anterior como predictor para el modelo Baseline

Métricas en entrenamiento y testeo

El modelo que mostró mejor rendimiento tuvo un entrenamiento total de 14,23 segundos y midiendo un *RMSE* de 2,930 °C al evaluarlo con el conjunto de datos de testeo. A su vez, le llevó 3,73 segundos calcular las predicciones para este conjunto. La Figura 4.2 muestra el descenso del error a medida que el modelo avanza en el entrenamiento (es decir, cada vez que se generaba un árbol nuevo).

Con el *RMSE* hallado en este último modelo se puede comprobar como el proceso de configuración de hiper-parámetros es de utilidad para encontrar un modelo que se adapte mejor a los datos, en comparación al creado al usar los parámetros por defecto; y superando aún más al resultado del modelo Baseline, indicado previamente.

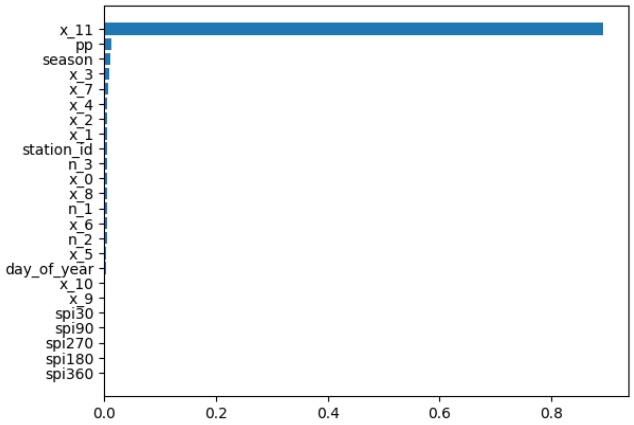

Figura 4.1: Gráfica de importancia de los parámetros de entrada del modelo *XGBoost*

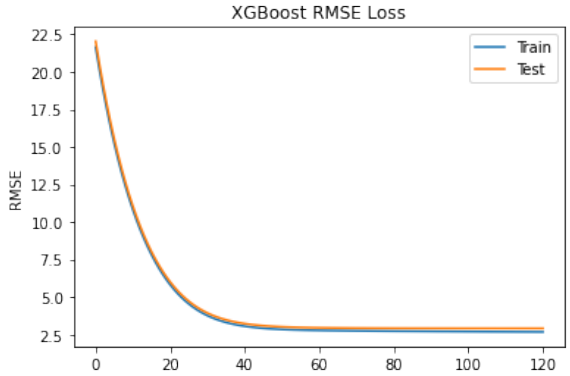

Figura 4.2: Gráfico de error en entrenamiento y test, a través de las iteraciones (es decir, cada vez que se generaba un árbol nuevo).

4.3. Graph WaveNet

4.3.1. Resultados intermedios

Al igual que en el caso de *XGBoost*, resulta interesante evaluar cómo es el desempeño de *Graph WaveNet* al realizarle un proceso de ingeniería de características en su información temporal y meteorológica.

Es relevante tener en cuenta cómo es el manejo de las dependencias espaciales del modelo detallado en la Sección 2.4.2. Mediante el uso de su matriz de adyacencia, el modelo siempre considera la información espacial del problema, por lo que este aspecto ya se encuentra cubierto. Para esta experimentación optamos ir con la matriz de adyacencia únicamente auto-adaptativa (*Adaptive-only*). Más

detalles sobre esta matriz se encuentran en la Sección 4.3.3.

Partimos entonces del conjunto de datos inicial al que llamaremos conjunto "básico", donde solo hacemos uso de la información de fecha (*date*, índice con formato `YYYY-MM-DD`) y temperatura máxima de los 12 días anteriores (*t_max*, medición en °C) al que se quiere predecir. En el Cuadro 4.4 podemos ver que utilizando este conjunto de datos, el modelo logra obtener un *RMSE* de 2.7358 °C al evaluar el conjunto de validación. Podemos ver entonces que ya sin aplicarle ningún procesamiento de ingeniería de características, el modelo logra obtener un menor *RMSE* que *XGBoost*.

Aplicamos entonces ingeniería de características y agregamos mayor información temporal al conjunto. Para esto, al igual que en el caso de *XGBoost*, procesamos los datos temporales que ya tenemos para poder obtener características que aporten información temporal de manera un poco más explicita. Introducimos entonces las nuevas características al conjunto de datos: estación del año (*season*, número del 0 al 3), número del día del año (*day*, número del 1 al 356) y mes del año (*month*, número del 1 al 12). En el mismo Cuadro 4.4, podemos ver que al agregar mayor información temporal, el modelo disminuye su *RMSE* de validación.

Finalmente, agregamos las precipitaciones (*pp*, medición en mm.) de cada día, introduciendo así mayor información meteorológica. Podemos ver como el modelo con este conjunto de datos obtiene su menor *RMSE* (2.620 °C) en el conjunto de validación.

Conjunto de datos	**T**	**S**	**RMSE**
básico	12	1	2.798 °C
info. temporal	12	1	2.742 °C
info. temporal y precipitaciones	12	1	2.641 °C

Cuadro 4.4: Desempeño de modelos con conjuntos de datos intermedios de *Graph WaveNet*, evaluando el conjunto de validación.

Por tanto, serán todos estos (*date*, *t_max*, *season*, *day*, *month*, *pp*), los datos de entrada que utilizaremos en el modelo final de *Graph WaveNet*.

4.3.2. Configuración de hiper-parámetros

Para el caso de *Graph WaveNet* realizamos la configuración de los hiper-parámetros en etapas, y consideramos los hiper-parámetros que creemos relevantes para el problema:

- *learning rate*;
- *epochs*;
- *weight decay*; y
- *dropout*.

Primero configuramos el parámetro *learning rate*, probamos con un conjunto de posibles valores y graficamos sus medidas de *loss* (*MAE*) y *RMSE* para los datos de validación. Observamos el comportamiento de estas gráficas para determinar el valor óptimo, la idea es obtener una curva que decrezca lo más uniformemente posible. Los tres mejores valores que obtenemos son los mostrados del Cuadro 4.5.

learning rate	***RMSE***
0,003	2.609 °C
0,005	2.610 °C
0,009	2.589 °C

Cuadro 4.5: Tres mejores *learning rates* y sus *RMSE* obtenidos en la configuración de hiper-parámetros, evaluando los datos de validación.

La siguiente etapa es la de configurar el *dropout*, *weight decay* y los *epochs*. Para los primeros dos, nos definimos combinaciones de valores posibles, construyendo una grilla de configuración en conjunto con los tres valores de *learning rate* definidos en la etapa anterior. De forma automatizada, entrenamos todos los modelos que surgen de todas las configuraciones posibles. A su vez, a través de estos entrenamientos configuramos también los *epochs*. Para esto, mejoramos el código original de los autores de *Graph WaveNet*, agregando un *early stopping* basado en cortar el entrenamiento cuando no se observa mejora luego de un intervalo de paciencia (definido en 40 pasos). Registramos esta cantidad de iteraciones como el *epochs* de cada configuración de hiper-parámetros. En este proceso de mejora fuimos asistidos por los colaboradores externos del trabajo: (Shleifer, McCreery, y Chitters, 2019). Debido a limitaciones de recursos, en todo el proceso no modificamos otro hiper-parámetro relevante, el *batch size*, utilizando su valor por defecto, 64. Una vez culminado el proceso, determinamos que la combinación de parámetros que mejor performa en el conjunto de validación, es la mostrada en el Cuadro 4.6. De ahora en más, denominaremos "modelo final" a esta configuración de hiper-parámetros de *Graph WaveNet*.

Parámetro	**Valor**
weight decay	0.0001
learning rate	0.005
dropout	0.3
epochs	168

Cuadro 4.6: Valores finales de hiper-parámetros de *Graph WaveNet* luego de su configuración.

4.3.3. Resultados con el modelo final

A continuación se muestran las métricas obtenidas de la experimentación con la configuración final de *Graph WaveNet*.

Métricas en entrenamiento

En la Figura 4.3 se puede ver el comportamiento de la medida de *loss* de la arquitectura *(MAE)* a lo largo de las iteraciones de entrenamiento. Podemos observar como la curva de la misma desciende y a partir de la iteración 10000 se mantiene acotado entre los valores 1.5 y 2.

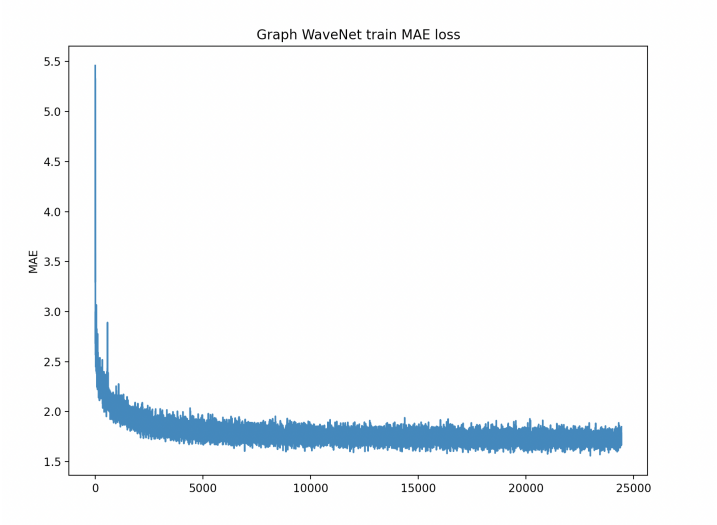

Figura 4.3: Gráfico de *loss* (*MAE*) en entrenamiento a través de las iteraciones para el modelo *Graph WaveNet*.

En la Figura 4.4 podemos observar el comportamiento del *RMSE* a lo largo de las iteraciones de entrenamiento. Vemos que el *RMSE* en términos generales, desciende y se mantiene acotado entre los valores 2 y 3. Sin embargo, existen claros picos donde el *RMSE* crece, lo que nos indicaría que el modelo comete errores de predicción para algunos puntos.

Resultados variando matrices de relación espacial

Como ya mencionamos en la Sección 2.4.2 del Capítulo del Estado del Arte, una de las características principales de *Graph WaveNet* es su capacidad de captar dependencias espaciales escondidas, mediante el uso de una matriz de adyacencia adaptativa. A continuación listamos los motivos por lo que se pueden usar las distintas configuraciones de las matrices:

- Si incluye o no información espacial del problema (Ecuaciones 2.15 y 2.14 respectivamente).
- En caso de incluir la información espacial, si esta es inicializada con datos reales del problema o con valores aleatorios.

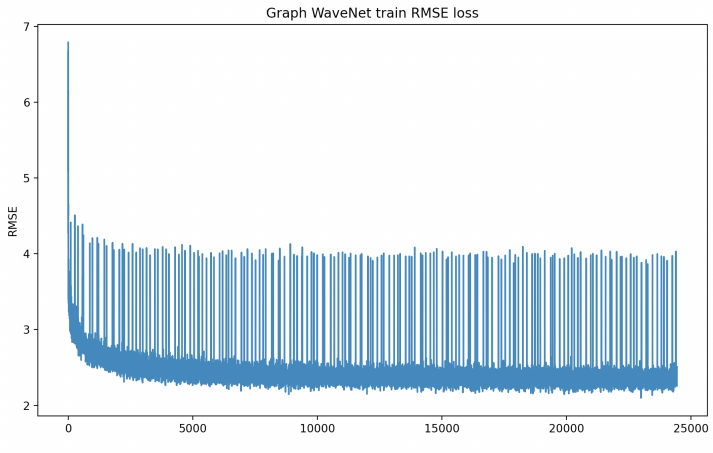

Figura 4.4: Gráfico de *RMSE* en entrenamiento a través de las iteraciones para el modelo *Graph WaveNet*.

- Si la matriz aprende o no aprende durante el entrenamiento.

Probamos el comportamiento del modelo al variar la configuración de su matriz de adyacencia. En esta experimentación probamos con 5 tipos distintos de matrices (ver Cuadro 4.7), de manera similar a la experimentación que hicieron los autores de *Graph WaveNet*. Como matriz de adyacencia original, seguimos el trabajo de los autores de la arquitectura, y creamos una matriz de adyacencia usando los datos de las distancias entre las estaciones de medición. Primero creamos una matriz de 137×137 cuyas entradas son las distancias en kilómetros entre cada par de estaciones. Luego normalizamos esta matriz y filtramos sus entradas, transformando en cero todo valor menor a 0.1, para evitar así que la matriz sea densa. La matriz que obtenemos de este proceso es la de la Figura 4.5. Esta matriz se utiliza en los casos en los que queremos inicializar la matriz de adyacencia con información de nuestro problema (los casos *Forward-only*, *Forward-backward*, y *Forward-backward-adaptive* del Cuadro 4.7).

Matriz	**Configuración**	***RMSE***
Identity	$[\boldsymbol{I}]$	3.0823 °C
Forward-only	$[\boldsymbol{P}]$	2.6851 °C
Adaptive-only	$[\tilde{\mathbf{A}}_{adp}]$	2.6160 °C
Forward-backward	$[\mathbf{P}_f, \mathbf{P}_b]$	2.6713 °C
Forward-backward-adaptive	$[\mathbf{P}_f, \mathbf{P}_b, \tilde{\mathbf{A}}_{adp}]$	2.6159 °C

Cuadro 4.7: Resultado de entrenar el modelo final de *Graph WaveNet* con sus distintas matrices de adyacencia.

En el Cuadro 4.7 se pueden ver los resultados del modelo al variar la configuración inicial de matriz de adyacencia, para los 5 tipos distintos estudiados.

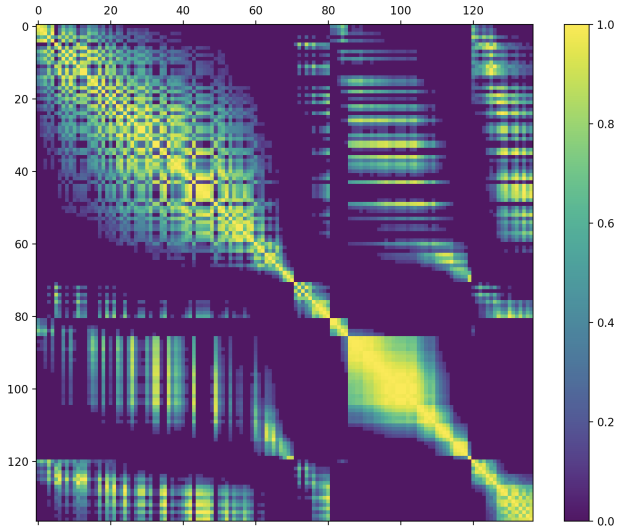

Figura 4.5: Gráfico de matriz de adyacencia de estaciones, para inicializar la matriz adaptativa de *Graph WaveNet*.

Como puede verse, la matriz que performa mejor es la *Forward Backward Adaptive* inicializada de forma aleatoria.

Algunas de las configuraciones hacen uso de la matriz auto-adaptativa $\tilde{\mathbf{A}}_{adp}$, la cual aprende las dependencias espaciales escondidas en nuestro problema. En la Figura 4.6 puede verse el resultado de esta matriz al final del entrenamiento para el caso de matriz de adyacencia *Forward-backward-adaptive*.

4.4. Comparación entre los tres modelos para la predicción del siguiente día

Resulta interesante comparar el desempeño de los modelos finales de *Graph WaveNet*, *XGBoost* y *Baseline*, para el conjunto de datos de testeo. En el Cuadro 4.8 se puede ver el *RMSE* y los tiempos de entrenamiento y testeo para los modelos para una ventana de 12 días en el pasado y una predicción del siguiente día.

El Cuadro 4.8 resume los resultados de los modelos finales de los respectivos procesos de configuración de hiper-parámetros y evaluación frente al conjunto de testeo para la tarea de predicción del día siguiente usando el histórico de 12 días. La columna $T = 12$ refiere a las métricas obtenidas por los modelos en dichas tareas. El modelo Baseline no registra tiempo de entrenamiento ni testeo, ya que la inferencia de los valores es inmediata a partir del último registro de los datos de entrada. Podemos apreciar que *Graph WaveNet* es el modelo que

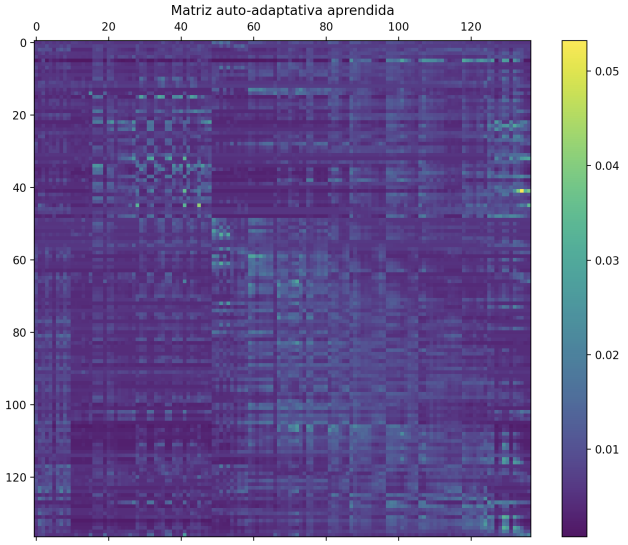

Figura 4.6: Gráfico de matriz auto-adaptativa $\tilde{\mathbf{A}}_{adp}$ al finalizar el entrenamiento de *Graph WaveNet* con configuración *Forward-backward-adaptive*.

		T = 12	**Tiempo de entrenamiento**	**Tiempo de testing**
RMSE	**XGBoost**	2.930 °C	14 s	3.73 s
	Graph WaveNet	2.544 °C	2889 s	3.36 s
	Baseline	3.508 °C	0.00 s	0.00 s

Cuadro 4.8: Comparación de desempeño entre *Graph WaveNet*, *XGBoost* y Baseline en el conjunto de datos de *test* para la predicción del siguiente día en función de los 12 días anteriores.

mejor performa por casi medio grado centígrado en comparación a *XGBoost* y casi un grado centígrado respecto a *Baseline*. Al considerar los tiempos de ejecución, podemos ver que *Graph WaveNet* es el modelo que requiere mayor tiempo de entrenamiento, lo que puede deberse a no encontrarse optimizada su implementación. Aun así, los tres modelos requieren de un corto tiempo para su entrenamiento.

Al comparar nuestros resultados con los modelos analizados en el estado del arte, que presentan características similares en términos de tipo de medición y objetivo predictivo, podemos notar que nuestros modelos obtuvieron métricas inferiores en comparación. Es decir, los modelos previamente estudiados mostraron un desempeño significativamente mejor que los nuestros.

La Figura 4.7 muestra las predicciones realizadas por los modelos para la fecha 05/07/2013, en la estación meteorológica de Artigas (Uruguay), allí se muestra el comportamiento de los datos de entrada (línea azul), la cual corresponde a las temperaturas máximas de dicha estación en los 12 días anteriores. x_0 corresponde al 24/06/2013 y x_11 al 04/07/2013. Podemos apreciar como *Graph WaveNet* es el modelo cuya predicción para esos datos de entrada es la más próxima al valor real, seguido en este caso por el *Baseline*.

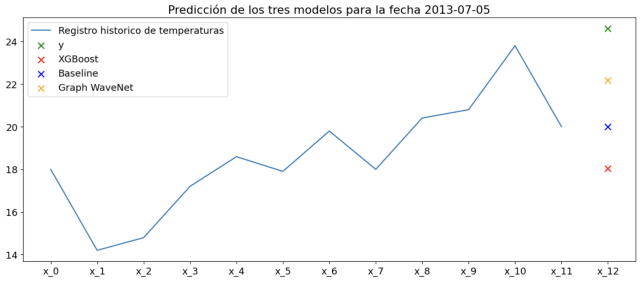

Figura 4.7: Gráfico de comparación de ventana de tiempo en el pasado y predicciones entre *XGBoost*, *Graph WaveNet*, *Baseline* y el valor real para la estación 11000 (Artigas, Uruguay) para la fecha 2013-07-05

Aunque esta imagen logra reflejar la secuencia de datos en la ventana de tiempo de entrada junto con la predicción correspondiente, es muy específico para poder comprender los comportamientos de los modelos. Las siguientes figuras buscan resumir los mismos.

En la Figura 4.8 podemos ver la comparación de las predicciones de los tres modelos con el valor real de temperatura en distintas estaciones meteorológicas y períodos de tiempo. Allí se puede apreciar como la curva del *Baseline* corresponde a su definición, ya que es exactamente igual al valor real, pero desplazada un día.

En el primer gráfico podemos notar como las predicciones de *Graph WaveNet* (línea verde) presenta similitudes con el valor real esperado, y aún más cuando ocurren temperaturas altas. Por otra parte, *XGBoost* (línea naranja) aparenta ser más conservador en la predicción de temperaturas y no logra acercarse tanto

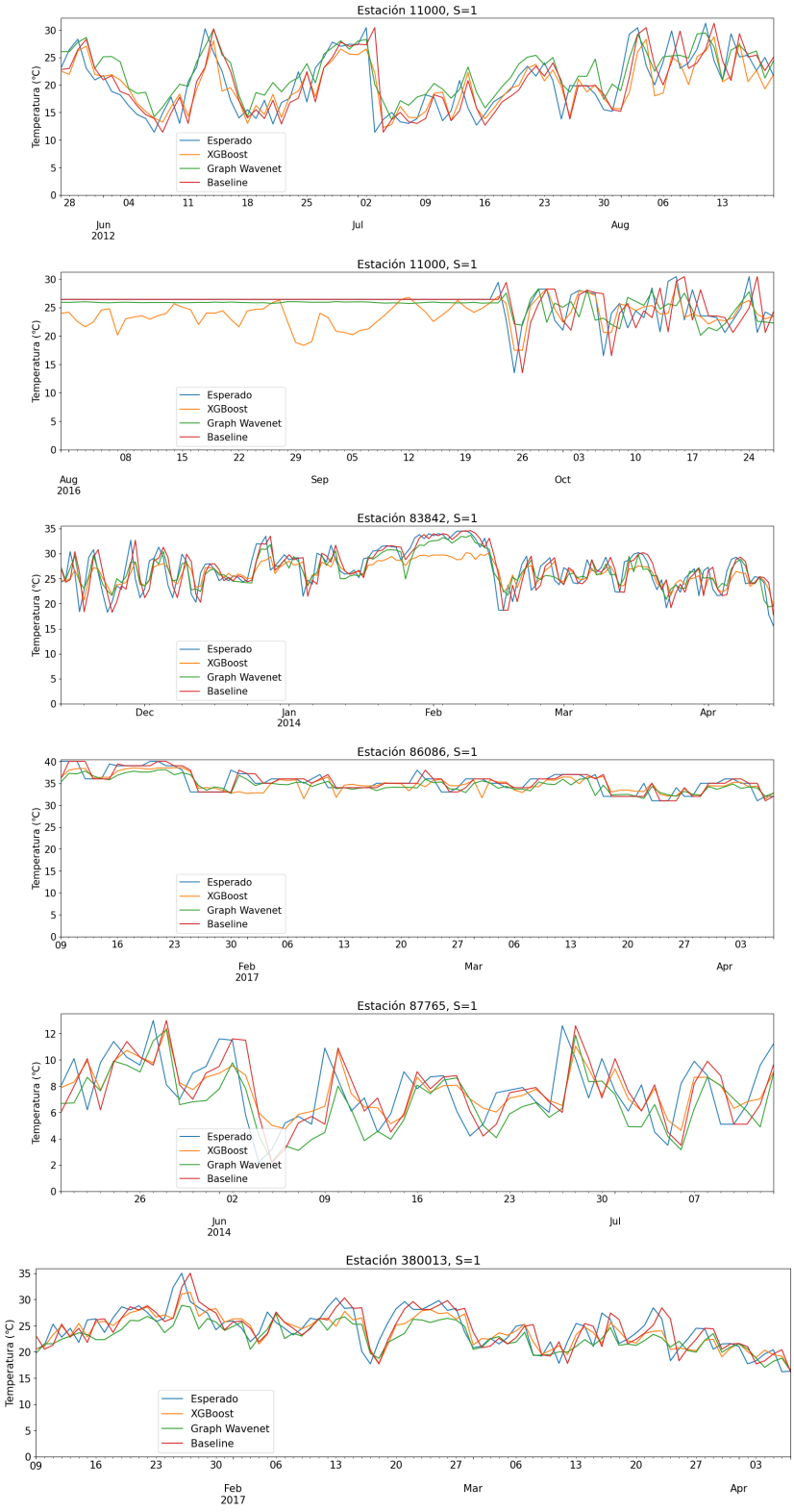

Figura 4.8: Gráficos de comparaciones de predicciones para el siguiente día (S=1) entre los modelos y el valor real en distintas estaciones y periodos de tiempo.

al valor real, principalmente cuando las temperaturas son altas.

El siguiente gráfico refleja las predicciones de los modelos para la misma estación pero en un período diferente. Podemos apreciar como al principio de la línea de tiempo nos encontramos con información faltante, presentándose todos los predictores (exceptuando *XGBoost*) con una línea recta, donde el valor real y el correspondiente al Baseline se encuentran superpuestos al no variar en el tiempo. Aquí se observa como *Graph WaveNet* infiere que la temperatura debe mantenerse constante, ya que la entrada lo indica; sin embargo, *XGBoost* presenta oscilaciones aunque los datos de entrada sean un valor constante; esto nos permite suponer que es capaz de tener memoria histórica basándose en información cronológica, además de los datos de entrada. Por ejemplo, puede estar ocurriendo que sea capaz de recordar un comportamiento de las temperaturas, o el valor registrado en los datos de entrada para esa época del año. Estas hipótesis se profundizará en la siguiente sección.

En las últimas 4 gráficas se puede ver que los modelos logran comportamientos similares, y en algunos casos, *XGBoost* se acerca mejor al valor real que *Graph WaveNet* (la quinta gráfica, en el mes de junio). Sin embargo, podemos apreciar que el modelo basado en grafos logra mejores predicciones en general, como también que se 'arriesga más' al momento de predecir una temperatura, pasando por encima o por debajo del valor real.

4.5. Comportamiento al variar las ventanas de tiempo

Una vez obtenidas las configuraciones finales de cada modelo y de haber comparado sus desempeños, resulta interesante experimentar sus capacidades de predicción a lo largo del tiempo. Por un lado, queremos evaluar el desempeño de los modelos al variar la cantidad de días de información del pasado que usan (ventana T del pasado) para la predicción de la temperatura máxima del día siguiente, por otro lado, queremos probar variar la cantidad de días de predicción de cada modelo (ventana S del futuro).

La bibliografía nos muestra que ya se han realizado experimentos al variar las ventanas de tiempo. No se indica de forma generalizada el tamaño de la ventana de tiempo de entrada; sin embargo, en lo que respecta a la salida, modelos con arquitecturas similares a las nuestras experimentan hasta 5 o 6 días hacia adelante. Estas predicciones pueden ser del conjunto total de días o solo el último de ellos. En todos los casos, la bibliografía confirma que este tipo de modelos comienzan a tener peor rendimiento cuando se buscan predicciones en conjunto o al buscar una predicción en un futuro más lejano.

Variando la ventana T del pasado

Queremos evaluar el desempeño de los modelos al variar la cantidad de días del pasado que usan como información de entrada para predecir la temperatura máxima del día siguiente. Para esto probamos con ventanas de tiempo (T) de

tamaño 2, 5, y 12. En el Cuadro 4.9 podemos ver el desempeño de los tres modelos al variar la ventana T del pasado. Podemos observar que *XGBoost* y *Graph WaveNet* obtienen un mejor desempeño al tener mayor información del pasado, ya que tienen un menor *RMSE* cuando T es de tamaño 12. *Graph WaveNet* tiene el mejor desempeño para los tres tamaños de ventana T, logrando su mejor desempeño cuando T es de tamaño 12.

		T = 2	**T = 5**	**T = 12**
RMSE	**XGBoost**	3.025 °C	3.010 °C	2.930 °C
	Graph WaveNet	2.687 °C	2.607 °C	2.544 °C
	Baseline	3.508 °C	3.508 °C	3.508 °C

Cuadro 4.9: Comparación de desempeño entre *Graph WaveNet*, *XGBoost* y *Baseline* al variar sus ventanas de tiempo en el pasado y predecir el siguiente día.

Variando la ventana S del futuro

Queremos también evaluar el desempeño de los modelos al variar la ventana de días de predicción de cada modelo (ventana S del futuro). En nuestro caso, mantendremos el tamaño de entrada ($T = 12$), y realizaremos distintos experimentos variando el tamaño de la ventana del futuro ($S = 1$, 3, 5 y 10), generando una predicción conjunta de todos los días en la ventana.

El Cuadro 4.10 compara el desempeño de los modelos para las distintas ventanas de tiempo en el futuro, calculando el promedio de la predicción conjunta. Es decir, cuando se predicen los siguientes 3 días ($S = 3$), se mide el error de la predicción de cada uno de ellos, y se toma el promedio de los 3 errores. En dicho Cuadro podemos observar como *XGBoost* muestra mejores resultados que *Graph WaveNet*, excepto cuando se predice el siguiente día. Con esto suponemos que el modelo basado en árboles de decisión performa mejor que *Graph WaveNet* cuando aumentamos el tamaño de la ventana de tiempo de predicción. Esta observación nos motiva a investigar las razones de esto, lo que profundizamos en las siguientes gráficas.

		S = 1	**S = 3**	**S = 5**	**S = 10**
RMSE	**XGBoost**	2.930 °C	3.522 °C	3.734 °C	3.922 °C
	Graph WaveNet	2.544 °C	3.818 °C	4.302 °C	4.059 °C
	Baseline	3.508 °C	5.201 °C	5.250 °C	5.284 °C

Cuadro 4.10: Comparación de desempeño entre *Graph WaveNet*, *XGBoost* y *Baseline* al variar sus ventanas de tiempo hacia adelante

Las gráficas de la Figura 4.9 representan las predicciones realizadas por los modelos de *XGBoost* y *Graph WaveNet* en los experimentos planteados. Las líneas representan las predicciones realizadas para el último día de la ventana establecida (con $S = 1$ se muestran las predicciones del día siguiente, cuando $S = 10$ se indican las predicciones del décimo día). Además de estas predicciones se pueden apreciar la medida real y dos predictores base: el promedio del conjunto

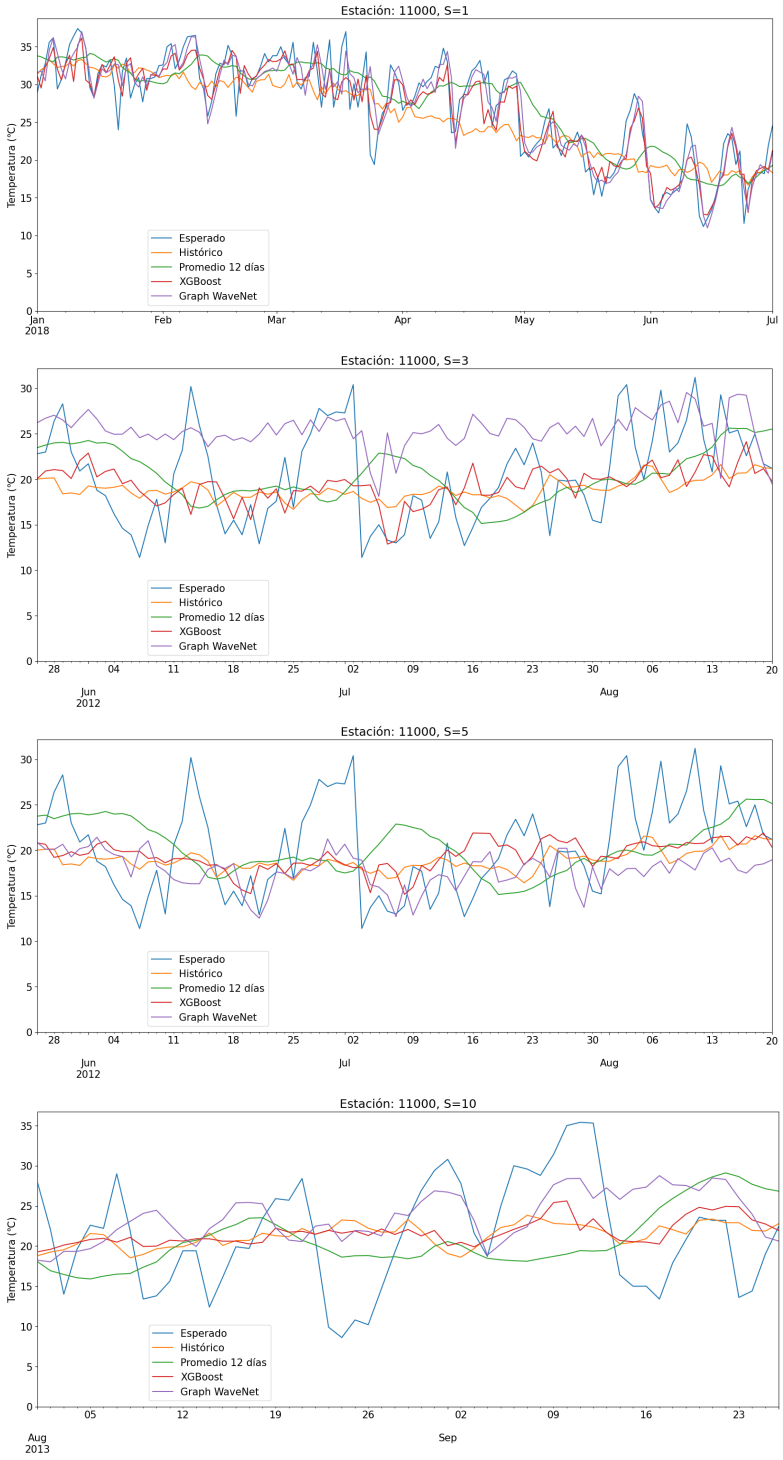

Figura 4.9: Comparación de predicciones al variar la ventana de predicción entre los modelos, el valor real y medidores auxiliares.

entrada y el promedio histórico de temperaturas registrado para la fecha en la estación donde se calcula la predicción.

Allí podemos apreciar como tanto *XGBoost* y *Graph WaveNet* se acercan bastante bien al valor real. Sin embargo, a medida que aumentamos la cantidad de días al futuro, las gráficas de los modelos comienzan a distanciarse del valor esperado, teniendo medidas más conservadoras. En particular, se puede apreciar como *XGBoost* comienza a aproximarse a la línea correspondiente al promedio histórico de cada fecha, teniendo en cuenta la estación de medición. Basándonos en esto último, podemos considerar que *XGBoost* tiende a devolver el promedio cuando se aumenta el tiempo de predicción, validando la idea planteada en la sección anterior con el ejemplo de un conjunto de entrada con temperatura constante.

Capítulo 5

Conclusiones y Trabajo Futuro

Finalmente, luego de pasar por los distintos capítulos, resumiendo las instancias que llevamos adelante en este trabajo, podemos indicar las conclusiones del mismo y también el posible trabajo a futuro en lo que respecta a esta línea de investigación.

5.1. Conclusiones

A lo largo de este proyecto trabajamos en la experimentación y comparación de dos enfoques distintos de aprendizaje automático. Comparamos una técnica de aprendizaje profundo moderna (*XGBoost*) y una técnica de aprendizaje profundo basada en grafos (*Graph WaveNet*); siendo nuestro problema objetivo el pronóstico meteorológico de temperaturas máximas usando información espacio-temporal. Es de resaltar que la utilización de técnicas avanzadas de aprendizaje automático en el área de meteorología, en general, es un trabajo incipiente a nivel mundial (Lam y cols., 2022) (Nguyen, Brandstetter, Kapoor, Gupta, y Grover, 2023). Particularmente, en la región no conocemos trabajos que hayan hecho uso de las técnicas utilizadas en nuestro trabajo para el problema planteado.

Comenzamos encaminándonos en nuestro objetivo de generar un *dataset* que centralizara la información meteorológica de la región, concluyendo que esto presentaba complejidades, tanto en el acceso a la información, como en la fiabilidad de los datos obtenidos. Así fue como procedimos a apoyarnos en expertos del área, participantes del equipo de trabajo de *ClimateDL*, para así consolidar datos meteorológicos de 5 países de la región. Este *dataset* completo, junto a su depuración y procesamiento, fueron esenciales para el desarrollo de nuestro trabajo, permitiéndonos explotar estos datos con el fin de optimizar las técnicas de aprendizaje utilizadas.

Elegimos dos técnicas novedosas de aprendizaje automático, *XGBoost* (un

modelo tradicional) y *Graph WaveNet* (una arquitectura basada en grafos). Gran parte de nuestro trabajo consistió en la configuración y optimización de estos modelos de predicción. Por un lado, *XGBoost* mostró ser una librería sencilla de entender y usar, con una comunidad muy activa y gran documentación. Sin embargo, al no manejar relaciones espacio-temporales por defecto, requirió un extenso proceso de *feature engineering* para agregar así características temporales, espaciales y meteorológicas, y lograr optimizar el desempeño del modelo. Este es un proceso, que en general, requiere de tiempo, recursos y un buen conocimiento del área de aplicación. Por otra parte, *Graph WaveNet* nos presentó dificultades de implementación, debiéndose a que es un modelo novedoso, con una comunidad muy pequeña, con muy poca documentación y escaso soporte. A diferencia de *XGBoost*, esta arquitectura sí tiene incorporada en su estructura el manejo de relaciones espacio-temporales, por lo que su etapa de *feature engineering* requirió significativamente menos esfuerzo. Asimismo, la arquitectura propone el uso de una matriz de adyacencia auto-adaptativa, la cual puede ser inicializada con información de adyacencia entre las estaciones de medición, y probar otras configuraciones útiles.

Ambos modelos requirieron una etapa de configuración de hiper-parámetros con el fin de buscar optimizar los resultados de los modelos. Esta fue una etapa larga que consumió tiempo y recursos, pero que mostró ser clave para conseguir las configuraciones de los modelos que logran mejor rendimiento.

Los modelos implementados obtuvieron mejores resultados que el modelo *Baseline* planteado (donde se utiliza como predicción de temperatura la última medida disponible). Esto ocurre para todas las versiones de los modelos, todos los tamaños de entrada, y todos los horizontes de días a predecir. En el caso de la predicción de temperatura del día siguiente en función de la información de los 12 días anteriores ($T = 12$ y $S = 1$), los modelos obtuvieron su mejor performance. El modelo *Baseline* obtuvo un *RMSE* de **3.508 °C**, mientras que *Graph WaveNet* y *XGBoost* obtuvieron **2.544 °C** y **2.930 °C** respectivamente, superando así el desempeño del *Baseline*. En el caso de la predicción de temperatura de los 10 días siguientes ($T = 12$ y $S = 10$), los modelos obtuvieron su peor desempeño. Sin embargo, el modelo *Baseline* obtuvo un *RMSE* de **5.284 °C**, mientras que *Graph WaveNet* y *XGBoost* obtuvieron **4.059 °C** y **3.922 °C** respectivamente, superando nuevamente el desempeño del *Baseline*. Esto nos demuestra de manera clara que el uso de metodologías de aprendizaje profundo logra buenos resultados de manera bastante directa.

Graph WaveNet resultó ser un modelo que logra buenos resultados sin mucha configuración. Para el caso de la predicción de la temperatura máxima del día siguiente en función de la información de los 12 días anteriores ($T = 12$ y $S = 1$), logra obtener mejores resultados que *XGBoost* sin requerir gran esfuerzo computacional ni humano. Esta arquitectura obtuvo un *RMSE* de **2.544 °C**, mientras que *XGBoost* obtuvo **2.930 °C**. Por lo que concluimos que *Graph WaveNet* logra buenos resultados para el caso de predicción de temperaturas próximas en función de las temperaturas históricas gracias a su manejo de la información espacio-temporal.

Para el caso de predecir temperaturas para una ventana hacia adelante mayor

a un día, si bien la diferencia no es tanta, *XGBoost* logra un mejor desempeño que *Graph WaveNet*. Esto podemos verlo para el caso de $T = 12$ y $S = 3$ donde *XGBoost* obtuvo un *RMSE* de **3.522 °C** a diferencia de *Graph WaveNet* que obtuvo un *RMSE* de **3.818 °C**. Lo mismo ocurre para el caso de $T = 12$ y $S = 5$, ya que *XGBoost* obtuvo un *RMSE* de **3.734 °C** a diferencia de *Graph WaveNet* que obtuvo un *RMSE* de **4.302 °C**. De manera similar, en el caso de $T = 12$ y $S = 10$, *XGBoost* obtuvo un *RMSE* de **3.922 °C** y *Graph WaveNet* obtuvo un *RMSE* de **4.059 °C**. Concluimos que esto ocurre debido a que el modelo *XGBoost* es capaz de identificar patrones a través del tiempo en la totalidad de los datos de entrenamiento. Estos patrones permiten acercar sus predicciones más a la media de los datos históricos, y, por lo tanto, obtener mejores resultados cuando se aumenta el horizonte de predicción y la incertidumbre del proceso. Por el contrario, *Graph WaveNet* se basa de manera más preponderante en la información que recibe de entrada para la predicción en específico, por lo que es capaz de predecir los casos donde la temperatura se mantiene constante.

La arquitectura de *Graph WaveNet* contiene su novedosa matriz de adyacencia adaptativa, la cual le permite aprender la información espacial del problema. La implementación del modelo permite varias configuraciones posibles de esta matriz, pudiendo ser inicializada con información espacial o de manera aleatoria. La matriz auto-adaptativa inicializada de manera aleatoria demostró ser la más efectiva, ya que obtuvo mejores resultados. La misma obtuvo un *RMSE* de **2.6160 °C**, a diferencia de la matriz inicializada con información espacial que obtuvo un *RMSE* de **2.6159 °C**.

Por otro lado, pudimos ver que ambos modelos son rápidos, tanto en sus tiempos de entrenamiento como en los de testeo. *XGBoost* muestra una ventaja, ya que logra mejores tiempos de entrenamiento que *Graph WaveNet* (debido a lo bien depurado y optimizado que se encuentra su código). Esto podemos verlo, por ejemplo, en el caso de $T = 12$ y $S = 1$, donde *XGBoost* logra entrenar en un tiempo de 14 segundos, a diferencia de *Graph WaveNet* que demora un total de 2889 segundos (aproximadamente 48 minutos).

En definitiva, salvando los problemas de implementación, *Graph WaveNet* parece ser la mejor solución para el problema de predicción a corto plazo. Por otro lado, *XGBoost* es superior cuando las ventanas de predicción crecen (S>1).

Como parte de nuestras contribuciones, en noviembre del 2022 participamos en el *workshop* de *ClimateDL*, donde presentamos al equipo los avances y hallazgos de nuestro trabajo hasta la fecha. Además, contribuimos en la publicación de una publicación académica (Marco y cols., 2023) que resume nuestro trabajo.

5.2. Trabajo Futuro

En el contexto de futuras investigaciones y desarrollos, se abren varias oportunidades para mejorar y expandir el alcance de este proyecto. En primer lugar, es posible seguir perfeccionando las implementaciones actuales, buscando mejorar el rendimiento de los modelos. Esto podría incluir la exploración de otro tipo de modelos o arquitecturas de aprendizaje automático tradicionales o basadas

en grafos. Para esto podríamos basarnos en trabajos similares del área.

En la implementación original del modelo *Graph WaveNet* desarrollada por sus autores, se empleó el *Error Absoluto Medio* (*MAE*) como métrica de ajuste durante el proceso de entrenamiento. Sin embargo, en nuestro estudio, optamos por utilizar el *Error Cuadrático Medio* (*RMSE*) como métrica principal tanto para la comparación de desempeño entre ambos modelos como en la función de pérdida de *XGBoost*. Esta elección se basó en que el *RMSE* es la métrica más utilizada en los trabajos relacionados. No obstante, debido a ciertas limitaciones técnicas encontradas en la implementación original propuesta por los autores, conservamos el *MAE* como función de pérdida en *Graph WaveNet*. Por lo que vemos interesante explorar la posibilidad de reemplazar el *MAE* por el *RMSE* como función de pérdida en *Graph WaveNet*. Tal cambio podría proporcionar una visión más coherente al comparar el desempeño de ambos modelos. Además, este cambio podría tener un impacto en la capacidad predictiva de *Graph WaveNet*, lo que abriría la puerta a mejoras en sus capacidades de predicción meteorológica.

A su vez, una cuestión importante es el tiempo de ejecución en el proceso de configuración de hiper-parámetros. Debido a las características y limitaciones en recursos de nuestro proyecto, optamos por limitar esta etapa. Sin embargo, posiblemente se puedan encontrar configuraciones de modelos con mejores desempeños si se amplía el rango de posibles valores para cada uno de los hiperparámetros, si se incorporan más hiper-parámetros para configurar, o si se brinda mayor tiempo de ejecución para explorar exhaustivamente estas opciones.

Otra consideración a tener en cuenta recae en los datos de entrenamiento. Aunque el conjunto de datos actual es de alta calidad, existe el potencial para obtener mejores resultados mediante cambios en este aspecto. Aumentar el tamaño del conjunto de datos, ya sea mediante la expansión temporal de los registros o la incorporación de nuevas estaciones de medición, podría ser beneficioso. Además, la inclusión de datos meteorológicos adicionales, como la velocidad del viento, la humedad y la temperatura mínima, podría mejorar aún más la capacidad predictiva.

Por otro lado, podríamos trabajar en la generación de características exógenas a partir de los datos existentes. Esta estrategia podría permitir a los modelos captar patrones subyacentes y relaciones complejas en los datos meteorológicos de manera más efectiva.

Para ampliar el alcance del proyecto, podríamos considerar un cambio en el objetivo de predicción. La exploración de la predicción de características distintas a la temperatura máxima, como la temperatura mínima o las precipitaciones, podría brindar nuevas perspectivas de investigación y aplicaciones prácticas. Una línea de investigación interesante podría ser la de evaluar la capacidad de los modelos para predecir múltiples características simultáneamente. Esta tarea desafiante podría llevar a soluciones más completas y útiles para aplicaciones meteorológicas.

Otra línea de investigación interesante sería la expansión del enfoque desde la meteorología hacia la climatología (ampliar sustancialmente el horizonte de predicción). Este cambio implicaría abordar un conjunto de problemas diferente

que se centra en el estudio a largo plazo de los patrones climáticos en lugar de las predicciones meteorológicas a corto plazo. En este nuevo contexto, deberíamos reconsiderar los modelos de aprendizaje utilizados y la selección de parámetros de entrada. La climatología involucra un análisis más amplio y a largo plazo de datos climáticos, lo que requeriría adaptar y desarrollar nuevos modelos de predicción.

En resumen, el futuro presenta diversas oportunidades para avanzar en esta área. La mejora continua del modelo, la expansión de datos y la exploración de objetivos de predicción alternativos son solo algunas de las direcciones que podrían conducir a avances significativos en la predicción meteorológica y sus aplicaciones relacionadas.

Referencias

Bauer, P., Thorpe, A., y Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, *525*(1476-4687), 47-55.

Beyer, M., Ahmad, R., Yang, B., y Rodríguez-Bocca, P. (2023). Deep spatial-temporal graph modeling for efficient ndvi forecasting. *Smart Agricultural Technology*, *4*, 100172. Descargado de https://www.sciencedirect.com/science/article/pii/S2772375523000023 doi: https://doi.org/10.1016/j.atech.2023.100172

Bi, Kaifeng and Xie, Lingxiand Zhang, Henghengand Chen, Xinand Gu, Xiaotaoand Tian, Qi, t. (2023, 01 de Jul). *Nature*, *619*(7970), 533-538. Descargado de https://doi.org/10.1038/s41586-023-06185-3 doi: 10.1038/s41586-023-06185-3

Chatfield, C. (2000). *Time-series forecasting (1st ed.)*. New York, NY: Chapman and Hall/CRC.

Chen, T., y Guestrin, C. (2016). Xgboost: A scalable tree boosting system. En *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (p. 785–794). New York, NY, USA: Association for Computing Machinery. Descargado de https://doi.org/10.1145/2939672.2939785 doi: 10.1145/2939672.2939785

Coleman, J., y Law, K. (2015). Meteorology. En *Reference module in earth systems and environmental sciences*. Elsevier. Descargado de https://www.sciencedirect.com/science/article/pii/B9780124095489094926 doi: https://doi.org/10.1016/B978-0-12-409548-9.09492-6

Cruz, B., Martínez, I. S., Abed, S. R., Ábalo, A. G., Lorenzo, R. G., y Matilde, M. (2007). Redes neuronales recurrentes para el análisis de secuencias. *Revista Cubana de Ciencias Informáticas*. Descargado de https://www.redalyc.org/articulo.oa?id=378343634004

Distributed (Deep) Machine Learning Community. (2023). *Categorical data*. https://xgboost.readthedocs.io/en/stable/index.html. (Accedido el: 2023-03-20)

Dorogush, A. V., Ershov, V., y Gulin, A. (2018). *Catboost: gradient boosting with categorical features support.*

Granger, C. W. J., y Joyeux, R. (1980). An introduction to long-memory time series models and fractional differencing. *Journal of Time Series Analysis*, *1*(1), 15-29.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354-377. Descargado de https://www.sciencedirect.com/science/article/pii/S0031320317304120 doi: https://doi.org/10.1016/j.patcog.2017.10.013

He, K., Zhang, X., Ren, S., y Sun, J. (2016). Deep residual learning for image recognition. En *2016 ieee conference on computer vision and pattern recognition (cvpr)* (p. 770-778). doi: 10.1109/CVPR.2016.90

Heaton, J. (2016). An empirical analysis of feature engineering for predictive modeling. En *Southeastcon 2016* (p. 1-6). doi: 10.1109/SECON.2016.7506650

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., ... Thépaut, J.-N. (2020). The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, *146*(730), 1999-2049. Descargado de https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3803 doi: https://doi.org/10.1002/qj.3803

Hochreiter, S., y Schmidhuber, J. (1997, 11). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780. Descargado de https://doi.org/10.1162/neco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Hutter, F., Kotthoff, L., y Vanschoren, J. (Eds.). (2019). *Automated machine learning - methods, systems, challenges*. Springer.

Jiang, W., y Luo, J. (2022, nov). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, *207*, 117921. Descargado de https://doi.org/10.1016%2Fj.eswa.2022.117921 doi: 10.1016/j.eswa.2022.117921

Ke, G., Meng, Q., Finely, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017, December). Lightgbm: A highly efficient gradient boosting decision tree. En *Advances in neural information processing systems 30 (nip 2017)*. Descargado de https://www.microsoft.com/en-us/research/publication/lightgbm-a-highly-efficient-gradient-boosting-decision-tree/

Keisler, R. (2022). Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*.

Kipf, T. N., y Welling, M. (2017). Semi-supervised classification with graph convolutional networks. En *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*. OpenReview.net.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., ... Battaglia, P. (2022). Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*. Descargado de https://doi.org/10.48550/arXiv.2212.12794

Li, Y., Yu, R., Shahabi, C., y Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. En *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings*. OpenReview.net.

Li, Y., Zhou, X., y Pan, M. (2022). Graph neural networks in urban intelli-

gence. En L. Wu, P. Cui, J. Pei, y L. Zhao (Eds.), *Graph neural networks: Foundations, frontiers, and applications* (pp. 579–593). Singapore: Springer Nature Singapore. Descargado de https://doi.org/10.1007/978-981-16-6054-2_27 doi: 10.1007/978-981-16-6054-2_27

Lim, B., y Zohren, S. (2021, feb). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194), 20200209. Descargado de https://doi.org/10.1098%2Frsta.2020.0209 doi: 10.1098/rsta.2020.0209

Lin, M.-L., Tsai, C. W., y Chen, C.-K. (2021). Daily maximum temperature forecasting in changing climate using a hybrid of multi-dimensional complementary ensemble empirical mode decomposition and radial basis function neural network. *Journal of Hydrology: Regional Studies*, *38*, 100923. Descargado de https://www.sciencedirect.com/science/article/pii/S221458182100152X doi: https://doi.org/10.1016/j.ejrh.2021.100923

Ma, J., Chan, J., Rajasegarar, S., y Leckie, C. (2022). Multi-attention graph neural networks for city-wide bus travel time estimation using limited data. *Expert Systems with Applications*, *202*, 117057. Descargado de https://www.sciencedirect.com/science/article/pii/S0957417422004717 doi: https://doi.org/10.1016/j.eswa.2022.117057

Ma, M., Xie, P., Teng, F., Wang, B., Ji, S., Zhang, J., y Li, T. (2023). Histgnn: Hierarchical spatio-temporal graph neural network for weather forecasting. *Information Sciences*, *648*, 119580. Descargado de https://www.sciencedirect.com/science/article/pii/S0020025523011659 doi: https://doi.org/10.1016/j.ins.2023.119580

Marco, G., Miranda, M. E., Rodríguez-Bocca, P., y Rubino, G. (2023). Maximal temperature forecasting under spatio-temporal interrelations using machine learning. En (p. 14). 2nd Workshop on Machine Learning for Irregular Time Series: Advances in Generative Models, Global Models and Self-Supervised Learning. Descargado de https://ml4its.github.io/ml4its2023/ (To appear)

Markudova, D., Baralis, E., Cagliero, L., Mellia, M., Vassio, L., Amparore, E. G., ... Salvatori, L. (2019). Heterogeneous industrial vehicle usage predictions: A real case. En P. Papotti (Ed.), *Proceedings of the workshops of the EDBT/ICDT 2019 joint conference, EDBT/ICDT 2019, lisbon, portugal, march 26, 2019* (Vol. 2322). CEUR-WS.org. Descargado de https://ceur-ws.org/Vol-2322/DARLIAP_13.pdf

Mehrkanoon, S. (2019, 06). Deep shared representation learning for weather elements forecasting. *Knowledge-Based Systems*, *179*, 120-128. Descargado de https://www.researchgate.net/publication/333747424_Deep_shared_representation_learning_for_weather_elements_forecasting/citation/download doi: 10.1016/j.knosys.2019.05.009

Nesmachnow, S., y Iturriaga, S. (2019). Cluster-uy: Collaborative scientific high performance computing in uruguay. En M. Torres y J. Klapp (Eds.),

Supercomputing (pp. 188–202). Cham: Springer International Publishing.

Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J. K., y Grover, A. (2023, enero). ClimaX: A foundation model for weather and climate. En (p. arXiv:2301.10343). doi: 10.48550/arXiv.2301.10343

Nyéki, A. and Kerepesi, C. and Daróczy, B. and Benczúr, A. and Milics, G. and Nagy, J. and Harsányi, E. and Kovács, A. J. and Neményi, M. (2021, 01 de Oct). Application of spatio-temporal data in site-specific maize yield prediction with machine learning methods. *Precision Agriculture*, *22*(5), 1397-1415. doi: 10.1007/s11119-021-09833-8

Onur Bilgin and Pawel Maka and Thomas Vergutz and Siamak Mehrkanoon. (2021). Tent: Tensorized encoder transformer for temperature forecasting. *arXiv preprint arXiv:2106.14742*.

Ramchoun, H., Amine, M., Idrissi, J., Ghanou, Y., y Ettaouil, M. (2016, 01). Multilayer perceptron: Architecture optimization and training. *International Journal of Interactive Multimedia and Artificial Inteligence*, *4*, 26-30. doi: 10.9781/ijimai.2016.415

Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., y Thuerey, N. (2020, nov). WeatherBench: A benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, *12*(11). Descargado de https://doi.org/10.1029%2F2020ms002203 doi: 10.1029/2020ms002203

Rasp, S., y Thuerey, N. (2021, feb). Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for WeatherBench. *Journal of Advances in Modeling Earth Systems*, *13*(2). Descargado de https://doi.org/10.1029%2F2020ms002405 doi: 10.1029/2020ms002405

Roy, D. (2020, 01). Forecasting the air temperature at a weather station using deep neural networks. *Procedia Computer Science*, *178*, 38-46. doi: 10 .1016/j.procs.2020.11.005

Schönauer, M., Prinz, R., Väätäinen, K., Astrup, R., Pszenny, D., Lindeman, H., y Jaeger, D. (2022). Spatifo-temporal prediction of soil moisture using soil maps, topographic indices and smap retrievals. *International Journal of Applied Earth Observation and Geoinformation*, *108*, 102730. doi: https://doi.org/10.1016/j.jag.2022.102730

Seger, C. (2018). An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing. En (p. 34). Digitala Vetenskapliga Arkivet. Descargado de https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1259073&dswid=-6832

SHI, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., y WOO, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. En C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, y R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28). Curran Associates, Inc. Descargado de https://proceedings.neurips.cc/paper_files/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf

Shleifer, S., McCreery, C., y Chitters, V. (2019). Incrementally improving

graph wavenet performance on traffic prediction. *arXiv preprint arXiv:1912.07390*, *abs/1912.07390*. Descargado de http://arxiv.org/abs/1912.07390

Sun, B., Sun, T., y Jiao, P. (2021). Spatio-temporal segmented traffic flow prediction with anprs data based on improved xgboost. *Journal of Advanced Transportation*, *2021*, 1–24. Descargado de https://www.hindawi.com/journals/jat/2021/5559562/

Svozil, D., Kvasnicka, V., y Pospichal, J. (1997). Introduction to multilayer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, *39*(1), 43-62. Descargado de https://www.sciencedirect.com/science/article/pii/S0169743997000610 doi: https://doi.org/10.1016/S0169-7439(97)00061-0

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. En *Proceedings of the 31st international conference on neural information processing systems* (p. 6000–6010). Red Hook, NY, USA: Curran Associates Inc.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., y Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, *32*(1), 4–24.

Wu, Z., Pan, S., Long, G., Jiang, J., y Zhang, C. (2019). Graph wavenet for deep spatial-temporal graph modeling. En S. Kraus (Ed.), *Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI 2019, macao, china, august 10-16, 2019* (pp. 1907–1913). ijcai.org. Descargado de https://doi.org/10.24963/ijcai.2019/264 doi: 10.24963/ijcai.2019/264

Yu, F., y Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. En Y. Bengio y Y. LeCun (Eds.), *4th international conference on learning representations, ICLR 2016, san juan, puerto rico, may 2-4, 2016, conference track proceedings*. Descargado de http://arxiv.org/abs/1511.07122